



Provided by the author(s) and University of Galway in accordance with publisher policies. Please cite the published version when available.

Title	Contributions to deep learning methodologies
Author(s)	Bazrafkan, Shabab
Publication Date	2018-10-26
Publisher	NUI Galway
Item record	http://hdl.handle.net/10379/14628

Downloaded 2024-05-15T06:26:18Z

Some rights reserved. For more information, please see the item record link above.



Contributions to Deep Learning Methodologies



Shabab Bazrafkan

College of Engineering and Informatics
National University of Ireland, Galway

This dissertation is submitted for the degree of
Doctor of Philosophy

Supervisor: Prof. Peter Corcoran

October 2018

“The art of knowing is knowing what to ignore.”

~Rumi

Table of contents

List of figures	xv
List of tables	xvii
Nomenclature	xix
1 Introduction	1
1.1 Literature Review	2
1.2 Summery Of the Contributions in This Thesis	4
1.2.1 Semi Parallel Deep Neural Networks (SPDNN)	5
1.2.2 Data Preparation	7
1.2.3 Deep Generative Models	8
1.3 Thesis Structure	10
1.4 List of the Publications	10
2 Deep Neural Networks	13
2.1 Neural Networks Building Blocks	14
2.1.1 Convolutional Layer	18
2.1.2 Pooling Layer	19
2.1.3 Unpooling Layer	20
2.1.4 Fully Connected Layers	21
2.2 Activation Functions	21
2.3 Loss Functions	23
2.4 Optimization Algorithms	24
2.5 Deep Neural Network Architectures	27
3 Contributions and Methodologies	29
3.1 Semi Parallel Deep Neural Networks	30
3.2 Data Preparation	36

3.2.1	Augmentation From an Expert Knowledge	37
3.2.2	Smart Augmentation	38
3.3	Deep Generative Models	41
3.3.1	Latent Space Mapping for Generation of Object Elements with Cor- responding Data Annotation	43
3.3.2	Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN)	47
3.3.3	Versatile Auxiliary Regressor with Generative Adversarial Network (VAR+GAN)	49
3.4	Conclusion	51
3.5	Future Works	54
References		55
Appendix A Pushing the AI Envelope: Merging deep networks to accelerate edge artificial intelligence in consumer electronics devices and systems		63
Appendix B Semi-Parallel Deep Neural Network (SPDNN) Hybrid Architecture, First Application on Depth from Monocular Camera		71
Appendix C Deep Learning for Facial Expression Recognition: A step closer to a SmartPhone that Knows your Moods		93
Appendix D An End to End Deep Neural Network for Iris Segmentation in Un- constraint Scenarios		99
Appendix E Smart Augmentation Learning an Optimal Data Augmentation Strat- egy		117
Appendix F Latent Space Mapping for Generation of Object Elements with Cor- responding Data Annotation		131
Appendix G Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN)		139
Appendix H Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN), Multi Class Scenarios		149

Appendix I Versatile Auxiliary Regressor with Generative Adversarial network (VAR+GAN)	159
---	------------

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 80,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Shabab Bazrafkan
October 2018

Acknowledgements

Foremost, I would like to express my sincere gratitude to my Ph.D. advisor Prof. Peter Corcoran for the continuous support of my Ph.D. study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study. Thanks, Peter!

I am forever indebted to Prof. Christopher Dainty for his constant feedback on my work. I truly appreciate him for being a friend, for providing me with a lot of research insights. I would also like to thank Dr. Petronel Bigioi and Dr. Alexandru Drimbarean for taking me as a part of the FotoNation family and involving me in some exciting projects.

I am grateful to all of those with whom I have had the pleasure to work during this and other related projects. I would be remiss if I did not thank my friends and colleagues at the C3Imaging group; Hossein Javidnia for being a wonderful friend, for all his support during my Ph.D. and the great time we spent together, especially in Las Vegas.

A special thanks to Dr. Shejin Thavalengal and Dr. Claudia Costache for their constant help and support throughout this journey.

Thanks to great friend and flatmate forever (GFFF) Tudor Nedelcu for being a supportive friend, flatmate and sharing tones of movies and Cuba Libre. Thanks to my other best flatmate Mick Walsh who joined us in the last year. Sorry for all the grease in the sink by the way.

Thanks to Viktor Varkarakis for all the wonderful times, laughs, and spicy chicken wings; Best mentee ever. Thanks to Aoife McDonagh for all the support in running sessions, and Adrian Ungureanu, Asma Khatoon and Anuradha Kar for all times we had lunch at FotoNation.

Thanks to my unforgettable friends, Joe Lemley and Snail Lemley for their constant support and the great time in Edinburgh and Porto. We definitely should start our tremoço business in Ireland.

Thanks to Ashkan Parsi and Saba Madani. All the memorable afternoons that we spent together in cafes. Sharing the moments, walks and discussions, made me feel home.

Great thanks to Carol Gallego, a wonderful teacher and friend for all the support in last 3 years. I would never forget the Westport trips, non-stop laughs and irish breakfast pizza in Joyces Headford.

Special thanks to Maeve McManus, Juli Baxter, Denisa Direrova, Tetyana Tyshkevych, Daisy Spencer and Catherine Fortune for all the fun and support. I owe them all big time for all those fun moments ;)

Thanks to Pat Fortune and Caroline Askins -my sister from other mother-. It was a pleasure to know these incredibly fun people and have a great time at "Wexico".

Thanks to Antonino Vespoli and Imma De Francesco which gave me the chance of trying true italian food and also all the fun times at 1, Ros Caoin, Roscam.

And above everything, throughout the entire journey, I have benefitted immensely from the support of my family and especially my parents for their love, inspiration and always being there when I needed. I never thank you enough for being there for me without a doubt. I also want to thank my brother and sister in law which were always supporting me in every stage of my life.

I would like to acknowledge Science Foundation Ireland and Xperi Ireland (FotoNation) for the generous funding for my Ph.D.

Abstract

In recent years the Deep Neural Networks (DNN) has been using widely in a big range of machine learning and data-mining purposes. This pattern recognition approach can handle highly nonlinear problems.

In this work, three main contributions to DNN are presented. 1- A method called Semi Parallel Deep Neural Networks (SPDNN) is introduced wherein several deep architectures are mixed and merged using graph contraction technique to take advantage of all the parent networks. 2- The importance of data is investigated in several attempts and an augmentation technique know as Smart Augmentation is presented. 3- To extract more information from a database, multiple works on Generative Adversarial Networks (GAN) are given wherein the joint distribution of data and its ground truth is approximated and in other projects conditional generators for classification and regression problems are trained and tested.

List of figures

1.1	The growth in number of GAN models since 2014.	4
2.1	The typical pipeline of model preparation in Deep Learning.	13
2.2	A typical ANN with a fully connected hidden layers.	15
2.3	(a) A biological neuron (illustrated by Kimberly Sowell, used with permission) and (b) an artificial neuron.	16
2.4	Visualization of the learned convolutional filters at different layers. (Copyright Movidius, used with permission.)	18
2.5	(a) A biological neuron (A 4-D filter maps one 3-D space to another 3-D space using convolutions.	19
2.6	A pooling operation reduces the size of the feature space.	20
2.7	A 2×2 unpooling operation with repeating values.	20
2.8	A 2×2 sparse unpooling operation.	21
2.9	A generic deep neural network, with convolutional (conv) and pooling layers followed by fully connected dense layers.	27
2.10	An autoencoder is the merged structure of the encoder and decoder in a model.	28
3.1	SPDNN workflow. This technique merges several networks using graph contraction method.	31
3.2	Left: three different networks designed for a specific binary classification task. Right: the graph correspond to each network. The properties for each node is written on the top of the nodes in the form (layer structure, distance from input).	33
3.3	Left: merge input nodes and out nodes in to single input and output. Right: apply the graph contraction to the graph on the left by merging nodes with the same label and removing the parallel vertices.	33

3.4	Labelling for the graph in figure 3.3 (right). The tuples on the top of each node is (layer structure, distance to output node). The nodes with the same properties will get the same label.	34
3.5	The graph contraction is applied to the graph shown in figure 3.4.	34
3.6	The graph can be turned back to the neural network using the layer structure and the connections.	35
3.7	Training workflow for work presented in [1] (Appendix B).	35
3.8	Output of four experiments for a road scene.	36
3.9	Augmentation workflow applied in [2] (Appendix D).	38
3.10	Segmentation map generated by the network for low quality iris images. . .	39
3.11	Segmentation map generated by the network for low quality iris images. . .	39
3.12	Smart Augmnetation technique explained in [3] (Appendix E). Black lines correspond to the forward propagation. Red line is backpropagation for NetB, green lines are backpropagation for NetA1 and blue lines are backpropagation for NetA2	40
3.13	Augmentation network merges several samples from a class to generate a new sample from the same class.	42
3.14	Training procedure presented in Appendix F.	45
3.15	The inference for generating samples alongside their corresponding aspects.	45
3.16	Randomly generated faces and their corresponding landmarks.	46
3.17	Randomly generated low-quality iris images and their corresponding segmentation maps.	46
3.18	ACGAN vs presented model.	47
3.19	Generator trained using the proposed method (VAC+GAN).	48
3.20	Samples drawn from conditional generator trained using proposed scheme (VAC+GAN) on MNIST dataset. each row corresponds to one class.	49
3.21	The error from the regression network backpropagates through the generator in VAR+GAN framework.	50
3.22	The cBiGAN framework merges CGAN and BiGAN approaches in a single model.	50
3.23	Generator outputs for proposed method (VAR+GAN) given particular landmarks	52

List of tables

2.1	Essential optimization methods for DNN training.	25
-----	--	----

Nomenclature

Roman Symbols

ADAM Adaptive Moment Estimation

AE AutoEncoder

AI Artificial Intelligence

ANN Artificial Neural Networks

CE Consumer Electronics

CNN Convolutional Neural Networks

ELU Exponential Linear Unit

FCDNN Fully Convolutional Deep Neural Network

GAN Generative Adversarial Network

GD Gradient Descent

GPU Graphics Processing Unit

MSE Mean Squared Error

NIR Near InfraRed

ReLU Rectified Linear Unit

RLReLU Randomized Leaky Rectified Linear Unit

RNN Recurrent Neural Network

SELU Scaled Exponential Linear Unit

SGD Stochastic Gradient Descent

SPDNN Semi Parallel Deep Neural Network

SReLU S-shaped Rectified Linear Unit

VAC+GAN Versatile Auxiliary Classifier with Generative Adversarial Network

VAE Variational Auto-Encoder

VAR+GAN Versatile Auxiliary Regressor with Generative Adversarial Network

Chapter 1

Introduction

In the last few years, we have witnessed an exponential growth in research activity into the advanced training of convolutional neural networks (CNNs), a field that has become known as deep learning [4–6]. This has been triggered by a combination of the availability of massive data sets, thanks in part to a corresponding growth in big data, and the arrival of new graphics-processing-unit (GPU)-based hardware that enables these large data sets to be processed in reasonable timescales. Suddenly, a wide variety of long-standing problems in machine learning, artificial intelligence, and computer vision have seen significant improvements, often sufficient to break through long-standing performance barriers [7]. Across multiple fields, these achievements have inspired the development of improved tools and methodologies leading to the even broader applicability of deep learning [8–10].

In the recent years deep learning tool kits [11–16] and techniques matured from tools that were mostly oriented toward researchers into easily used product-enabling technology that can be used to add intelligence to almost any consumer device, even by nonexperts. We expect to see an explosion in products that take advantage of these resources in the coming years, with early adopters differentiating themselves from competitors and further refinement of technology and deep learning methods. Artificial neural networks (ANNs) are able to learn something about what they see and then can generalize that knowledge to examples (or samples) that they have never seen before [17]. This is a very powerful capability that humans often take for granted because our brains automatically do it so well. You are able to understand the concept of a rock after seeing and perhaps touching very few examples of rocks. From that point on, you can identify any rock, even those that are shaped differently or have different colors or textures from the rocks you've seen before. This approach is very different from the traditional method of teaching or explicitly programming computers based on detailed rules that must cover every possible outcome [4, 5]. The process of discerning

the category to which a piece of data belongs is called a classification task; one of the more famous uses of this technique is that of training a neural network. The ability to classify unseen examples is referred to as generalization. Not surprisingly, ANNs are especially powerful in tasks for which the appropriate outcome cannot be determined beforehand and thus cannot use traditional preprogrammed rules [18].

1.1 Literature Review

2012 is when the Deep Learning wave started with AlexNet [19], the winner of the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). The model achieved 15.4% top 5 test error while the next best algorithm stands at 26.2%. The wave continued by introducing the ZF Net [20] in 2013 which again won the ILSVRC with 11.2% error rate. The ZF Net is very similar to AlexNet with the difference that the number of channels increases after each pooling. But the main contribution of [20] is the new way of visualization that was introduced to investigate the features which are extracted in each layer of the network. This method is known as DeConvNet. Thanks to this approach it has been shown that the network is extracting low-level features at the early stages and while going deeper the filters are excited by higher level shapes presented in the input. This understanding has led researchers to find more elaborated applications for DNN such as style transfer [21, 22], wherein the extracted features in different layers of a pre-trained network (usually VGG 16 or VGG 19) are utilized to inject different styles into an image.

In 2014 a deep network was proposed known as VGG Net [23] with 19 layers wherein the convolutional layers are strictly 3×3 kernels followed by max-pooling layers. (For more information on convolution and pooling layers see section 2.1). In order to keep the information flow inside the network, the number of channels are doubled after each pooling layer. VGG net brought simplicity with depth to the DL science [24]. VGG Net did not win the ILSVRC in 2014, the reason is by that time the big names were already investing time and human resources on DL. GoogleNet [25] won the competition with a top 5 error rate of 6.7%. This network is a more than 100 layer DNN which is taking advantage of inception units. These units give the opportunity of having pooling and convolution at the same layer by placing them in parallel. There are several 1×1 , 3×3 and 5×5 convolutional layers and 3×3 pooling layer in each inception module. Despite its depth, GoogleNet has 12x fewer parameters compared to AlexNet.

Google was not the only big name interested in DL, in 2015 Microsoft's ResNet [26] acquired 3.6% error rate in ILSVRC which is almost twice better than humans. In fact, 2015 was the year that the machine beat up human in object recognition accuracy. The ResNet is made of several residual blocks wherein the input data goes through several convolutional layers and then adds up with the original data. The authors in [26] believe that optimizing the residual mapping is easier than optimizing the original data.

The other high impact work in DL science in recent years is the Region Based CNNs (RCNN) [27–29]. In these series of works the network provides a bounding box around objects inside the image and specifies a tag for each box by getting help from an object recognition module (AlexNet in this case). The works presented in Fast RCNN [28] and Faster RCNN [29] is obviously around speeding up the original idea for real-time use cases. After 2015 several object detection networks have been introduced including, Region-based Fully Convolutional Network (R-FCN) [30], You Only Look Once (YOLO) [31], Single-Shot Detector (SSD) [32], YOLO9000 and YOLOv2 [33], Neural Architecture Search Net (NASNet) [34], and Mask Region-based Convolutional Network (Mask R-CNN) [35].

Semantic segmentation is another important application of DNN. SegNet [36, 37] is the most celebrated semantic segmentation DNN and the reason is the ease of implementation alongside with its high-quality outputs. SegNet uses the convolutional layers of the VGG16 network as the encoder of the network and eliminates the fully connected layers, thus reducing the number of trainable parameters from 134M to 14.7M, which represents a reduction of 90% in the number of parameters to be trained. As most of the spatial resolution information is lost in the max-pooling operation, saving the information of the max-pooling indices and using this information in the decoder part of the network preserves the high-frequency information.

Classification and Regression are not the only problems that DL has a solution for. In 2014, DNNs has been utilized in estimation theory by the introduction of Generative Adversarial Networks (GAN) [38]. GANs are successful implementations of deep generative models, and there are multiple variations such as WGAN [39], EBGAN [40], BEGAN [41], ACGAN [42], and DCGAN [43], which have evolved from the original GAN by altering the loss function and/or the network architecture. A research [44] shows the exponential growth of new GAN related models in the literature in the last few years. (see figure 1.1¹).

¹https://github.com/hindupuravinash/the-gan-zoo/blob/master/cumulative_gans.jpg

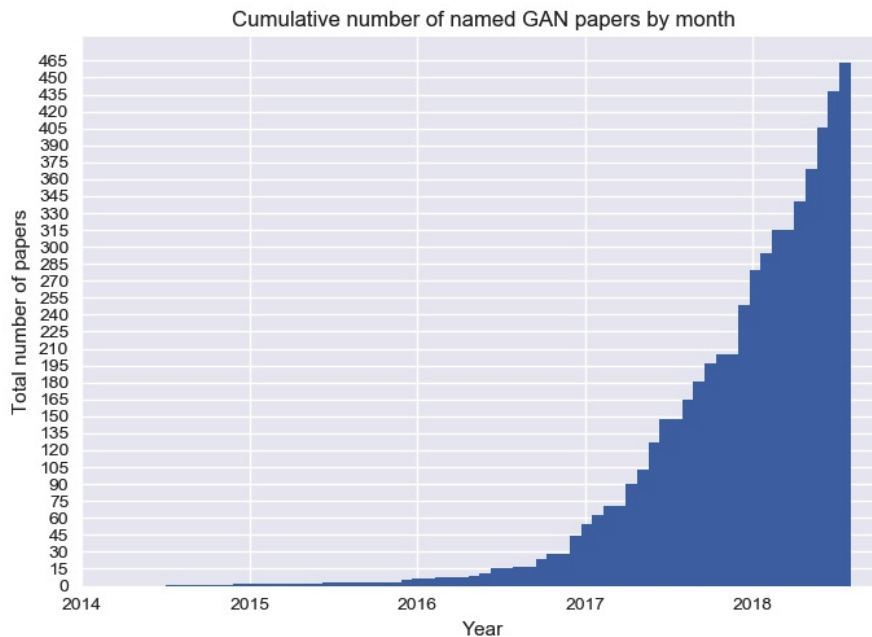


Fig. 1.1 The growth in number of GAN models since 2014.

1.2 Summery Of the Contributions in This Thesis

Although the Deep Neural Networks gave promising results in ILSVRC 2012-2014 and few groups in big companies were working on developing this technique, back in 2015 most of the scientists and engineers considered this solution as "Smoke and Mirrors". The main reason was the lack of analytical explanation of data reorientation inside the network in the training stage. A problem which stayed unsolved while writing these words.

Starting Deep Learning was one of the most challenging decisions in my career since in 2015 there was still a public unacceptability against DL. The main support for DL techniques is its excellent outcomes. The results are exceptionally better than other methods in a way that the Machine Learning community turned into using DL without asking for the analytical explanations.

I started to work on deep learning methods in early 2016. Contributions of this thesis is divided into three main fields:

1. Semi Parallel Deep Neural Networks (SPDNN).
2. Contributions to Data Preparation.

3. Contributions to Deep Generative Models.

In the following sections, these contributions are explained alongside with their corresponding publications.

1.2.1 Semi Parallel Deep Neural Networks (SPDNN)

This method was one of the earliest works of the current Ph.D. thesis. The main idea started while working on the iris segmentation project for low-quality images. Several networks trained on the iris database and in the test stage, each of the networks suffered from a set of specific artifacts. The solution was to put all these networks in parallel and train them together. In the test stage, we realized that each network compensated the artifact of other networks and the quality of the output was higher than each of networks individually. The next step was to find a way to get rid of similar layers in the parallel model to reduce the redundancy and size of the network. The proposed SPDNN method solves this problem by translating the neural network into a graph and apply graph contraction to every node twice. More information on this method is given in section 3.1. There are three main publications featuring SPDNN as follows:

Pushing the AI envelope: merging deep networks to accelerate edge artificial intelligence in consumer electronics devices and systems [45]

Deep neural networks (DNNs) are widely used by both academic and industry researchers to solve many long-standing problems in machine learning. There has been such a growth of research in this field, and it has been applied to so many varying problems, that it would be accurate to say that we may be living through the precursor of the singularity. But regardless of one's views on artificial intelligence (AI), there is no doubt that there is a wealth of recent research that leverages the use of various DNNs to solve a broad range of pattern recognition and classification problems. Examples range from the introduction of smart speakers with intelligent assistants to the application of DNNs to solve recalcitrant problems in computer vision for autonomous vehicles. Many of these problems can have very useful applications in the design of smarter consumer electronics (CE) systems and devices. The question for CE engineers is how to leverage this wealth of academic and industry research efforts, turning them into practical DNN solutions suitable for deployment in practical devices and electronic systems. In this work, an approach to merge and mix several neural networks into a single design is presented which is helping engineers to construct deep neural networks with lesser parameters while maintaining the same performance.

An End to End Deep Neural Network for Iris Segmentation in Unconstraint Scenarios [2]

With the increasing imaging and processing capabilities of today's mobile devices, user authentication using iris biometrics has become feasible. However, as the acquisition conditions become more unconstrained and as image quality is typically lower than dedicated iris acquisition systems, the accurate segmentation of iris regions is crucial for these devices. In this work, an end to end Fully Convolutional Deep Neural Network (FCDNN) design is proposed to perform the iris segmentation task for lower-quality iris images. The network design process is explained in detail, and the resulting network is trained and tuned using several large public iris datasets. Comprehensive inter-database comparisons are provided together with results from a selection of experiments detailing the effects of different tunings of the network. Finally, the proposed model is compared with SegNet-basic, and a near-optimal tuning of the network is compared to a selection of other state-of-art iris segmentation algorithms. The results show very promising performance from the optimized Deep Neural Networks design when compared with state-of-art techniques applied to the same lower quality datasets.

Semi-Parallel Deep Neural Network (SPDNN) Hybrid Architecture, First Application on Depth from Monocular Camera [1]

Deep neural networks are applied to a wide range of problems in recent years. In this work, Convolutional Neural Network (CNN) is applied to the problem of determining the depth from a single camera image (monocular depth). Eight different networks are designed to perform depth estimation, each of them suitable for a feature level. Networks with different pooling sizes determine different feature levels. After designing a set of networks, these models may be combined into a single network topology using graph optimization techniques. This "Semi Parallel Deep Neural Network (SPDNN)" eliminates duplicated common network layers, and can be further optimized by retraining to achieve an improved model compared to the individual topologies. In this study, four SPDNN models are trained and have been evaluated at 2 stages on the KITTI dataset. The ground truth images in the first part of the experiment are provided by the benchmark, and for the second part, the ground truth images are the depth map results from applying a state-of-the-art stereo matching method. The results of this evaluation demonstrate that using post-processing techniques to refine the target of the network increases the accuracy of depth estimation on individual mono images. The second evaluation shows that using segmentation data alongside the original data as the

input can improve the depth estimation results to a point where performance is comparable with stereo depth estimation. The computational time is also discussed in this study.

1.2.2 Data Preparation

The biggest lesson of my Ph.D. was to understand the actual value of the data in machine learning. Data is the most important part of any data-driven method including deep learning. It determines the structure of the network, the depth of the model and any other hyperparameter. The first project of deep learning during my Ph.D. was the facial expression classification using deep learning. In this work, the inter-database evaluations show the importance of mixing and merging databases in the generalization of the model.

Another contribution of this thesis is the data augmentation techniques which is a game-changing factor in providing higher quality models. There are three main publications on Data Preparation as follows:

Deep Learning for Facial Expression Recognition: A step closer to a SmartPhone that Knows your Moods [46]

By growing the capacity and processing power of the handheld devices nowadays, a wide range of capabilities can be implemented in these devices to make them more intelligent and user friendly. Determining the mood of the user can be used in order to provide suitable reactions from the device in different conditions. One of the most studied ways of mood detection is by using facial expressions, which is still one of the challenging fields in pattern recognition and machine learning science. Deep Neural Networks (DNN) have been widely used in order to overcome the difficulties in facial expression classification. In this paper it is shown that the classification accuracy is significantly lower when the network is trained with one database and tested with a different database. A solution for obtaining a general and robust network is given as well.

An End to End Deep Neural Network for Iris Segmentation in Unconstraint Scenarios [2]

In this work, the SPDNN method have been used to design the DNN and a set of methods to generate and augment suitable lower quality iris images from the high-quality public databases are provided. The network is trained on Near InfraRed (NIR) images initially and later tuned on additional datasets derived from visible images. These agumentation steps include contrast reduction, shadowing and motion blurring. The results show the superior results of the proposed method compared to the state of the art methods in literature.

Smart Augmentation Learning an Optimal Data Augmentation Strategy [3]

A recurring problem faced when training neural networks is that there is typically not enough data to maximize the generalization capability of deep neural networks. There are many techniques to address this, including data augmentation, dropout, and transfer learning. In this paper, we introduce an additional method, which we call smart augmentation and we show how to use it to increase the accuracy and reduce over fitting on a target network. Smart augmentation works, by creating a network that learns how to generate augmented data during the training process of a target network in a way that reduces that network's loss. This allows us to learn augmentations that minimize the error of that network. Smart augmentation has shown the potential to increase accuracy by demonstrably significant measures on all data sets tested. In addition, it has shown potential to achieve similar or improved performance levels with significantly smaller network sizes in a number of tested cases.

1.2.3 Deep Generative Models

Generative Adversarial Networks (GAN) introduced DL to the estimation theory. In this method, a DNN learns the data distribution and is able to draw random samples from the same distribution. The contributions to GANs are mainly based on learning the latent space for a pre-trained GAN which helps to learn the mutual distribution of the data with any aspect of it. And the other work is to map the aspect space to latent space in order to draw random samples with specific aspects.

Four main publications on Deep Generative Models are as follows:

Latent Space Mapping for Generation of Object Elements with Corresponding Data Annotation

Deep neural generative models such as Variational Auto-Encoders (VAE) and Generative Adversarial Networks (GAN) are giving promising results in estimating the data distribution across a range of machine learning fields of application. Recent results have been especially impressive in image synthesis where learning the spatial appearance information is a key goal. This enables the generation of intermediate spatial data that corresponds to the original dataset. In the training stage, these models learn to decrease the distance of their output distribution to the actual data, and in the test phase, they map a latent space to the data space. Since these models have already learned their latent space mapping, one question is whether there is a function mapping the latent space to any aspect of the database for the given generator. In this work, it has been shown that that this mapping is relatively straightforward using small neural network models and by minimizing the mean square error.

As a demonstration of this technique, two example use cases have been implemented: firstly, the idea to generate facial images with corresponding landmark data, and secondly generation of low-quality iris images (as would be captured with a smartphone userfacing camera) with a corresponding ground-truth segmentation contour.

Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN) [47]

One of the most interesting challenges in Artificial Intelligence is to train conditional generators which are able to provide labeled fake samples drawn from a specific distribution. In this work, a new framework is presented to train a deep conditional generator by placing a classifier in parallel with the discriminator and back propagate the classification error through the generator network. The method is versatile and is applicable to any variations of Generative Adversarial Network (GAN) implementation, and also is giving superior results compare to similar methods.

Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN), Multi Class Scenarios [48]

Conditional generators learn the data distribution for each class in a multi-class scenario and generate samples for a specific class given the right input from the latent space. In this work, a method known as "Versatile Auxiliary Classifier with Generative Adversarial Network" for multi-class scenarios is presented. In this technique, the Generative Adversarial Networks (GAN)'s generator is turned into a conditional generator by placing a multi-class classifier in parallel with the discriminator network and backpropagate the classification error through the generator. This technique is versatile enough to be applied to any GAN implementation. The results on two databases and comparisons with other method are provided as well.

Versatile Auxiliary Regressor with Generative Adversarial network (VAR+GAN) [49]

Being able to generate constrained samples is one of the most appealing applications of the deep generators. Conditional generators are one of the successful implementations of such models wherein the created samples are constrained to a specific class. In this work, the application of these networks is extended to regression problems wherein the conditional generator is restrained to any continuous aspect of the data. A new loss function is presented for the regression network and also implementations for generating faces with any particular set of landmarks is provided.

1.3 Thesis Structure

Next chapter, provides a detailed explanation of Deep Neural Networks including building elements, activation functions, and optimization techniques. And the third chapter presents the contributions on Deep Learning including network design, database preparation, and deep generative models. Appendices A to I contain the main contributions to the literature on DL science.

1.4 List of the Publications

SPDNN

1. **S. Bazrafkan**, and P. Corcoran, "PUSHING THE AI ENVELOPE: MERGING DEEP NETWORKS TO ACCELERATE EDGE ARTIFICIAL INTELLIGENCE IN CONSUMER ELECTRONICS DEVICES AND SYSTEMS", *IEEE Consumer Electronics Magazine* 7 (2), 55-61
2. **S. Bazrafkan**, S. Thavalengal, and P. Corcoran, "AN END TO END DEEP NEURAL NETWORK FOR IRIS SEGMENTATION IN UNCONSTRAINED SCENARIOS", *Neural Networks*, vol. 106, pp.79 – 95, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S089360801830193X>
3. **S. Bazrafkan**, H. Javidnia, J. Lemley, and P. Corcoran, "SEMIPARALLEL DEEP NEURAL NETWORK HYBRID ARCHITECTURE: FIRST APPLICATION ON DEPTH FROM MONOCULAR CAMERA", *Journal of Electronic Imaging*, vol. 27, pp. 27 – 47, 2018. [Online]. Available: <https://doi.org/10.1117/1.JEI.27.4.043041>
4. **S. Bazrafkan**, and P. Corcoran, "ENHANCING IRIS AUTHENTICATION ON HANDHELD DEVICES USING DEEP LEARNING DERIVED SEGMENTATION TECHNIQUES", *IEEE International Conference on Consumer Electronics*, (ICCE 2018)
5. AS. Ungureanu, **S. Bazrafkan**, and P. Corcoran, "DEEP LEARNING FOR HAND SEGMENTATION IN COMPLEX BACKGROUNDS", *IEEE International Conference on Consumer Electronics*, (ICCE 2018)

Data Preparation

1. **S. Bazrafkan**, T. Nedelcu, P. Filipczuk, and P. Corcoran, "DEEP LEARNING FOR FACIAL EXPRESSION RECOGNITION: A STEP CLOSER TO A SMARTPHONE THAT

- KNOWS YOUR MOODS," in IEEE International Conference on Consumer Electronics (ICCE), 2017.
2. **S. Bazrafkan**, S. Thavalengal, and P. Corcoran, "AN END TO END DEEP NEURAL NETWORK FOR IRIS SEGMENTATION IN UNCONSTRAINED SCENARIOS", arXiv preprint, arXiv:1712.02877. Accepted to be published at the Neural Networks journal, Elsevier.
 3. J. Lemley, **S. Bazrafkan**, and P. Corcoran, "SMART AUGMENTATION: LEARNING AN OPTIMAL DATA AUGMENTATION STRATEGY" IEEE Access, vol. 5. pp. 5858-5869, 2017.
 4. J. Lemley, **S. Bazrafkan**, and P. Corcoran, "LEARNING DATA AUGMENTATION FOR CONSUMER DEVICES AND SERVICES", IEEE International Conference on Consumer Electronics, (ICCE 2018)

Deep Generative Models

1. **S. Bazrafkan**, and P. Corcoran, "VERSATILE AUXILIARY REGRESSOR WITH GENERATIVE ADVERSARIAL NETWORK (VAR+GAN)", arXiv preprint, arXiv:1805.10864.
2. **S. Bazrafkan**, H. Javidnia, and P. Corcoran, "VERSATILE AUXILIARY CLASSIFIER WITH GENERATIVE ADVERSARIAL NETWORK (VAC+GAN)", arXiv preprint, arXiv:1805.00316, 2018.
3. **S. Bazrafkan**, and P. Corcoran, "VERSATILE AUXILIARY CLASSIFIER WITH GENERATIVE ADVERSARIAL NETWORK (VAC+GAN), MULTI CLASS SCENARIOS", arXiv preprint, arXiv:1806.07751, 2018.
4. **S. Bazrafkan**, H. Javidnia, and P. Corcoran, "FACE SYNTHESIS WITH LANDMARK POINTS FROM GENERATIVE ADVERSARIAL NETWORKS AND INVERSE LATENT SPACE MAPPING", arXiv preprint, arXiv:1802.00390.

Other Works: Related to Neural Networks

1. J. Lemley, **S. Bazrafkan**, and P. Corcoran, "DEEP LEARNING FOR CONSUMER DEVICES AND SERVICES: PUSHING THE LIMITS FOR MACHINE LEARNING, ARTIFICIAL INTELLIGENCE, AND COMPUTER VISION.," IEEE Consumer Electronics Magazine, vol. 6, no. 2. pp. 48-56, 2017.

2. **S. Bazrafkan**, J. Lemley, and P. Corcoran, "HYPER NEURONS?: ONE NEURON WITH INFINITE STATES," 26th Int. Conf. Artif. Neural Networks, 2017.
3. H. Javidnia, **S. Bazrafkan**, and P. Corcoran, "THE APPLICATION OF DEEP LEARNING ON DEPTH FROM MULTI-ARRAY CAMERA", IEEE International Conference on Consumer Electronics, (ICCE 2018)
4. J. Lemley, **S. Bazrafkan**, and P. Corcoran, "TRANSFER LEARNING OF TEMPORAL INFORMATION FOR DRIVER ACTION CLASSIFICATION," in The 28th Modern Artificial Intelligence and Cognitive Science Conference (MAICS), 2017.

Other Works: Not Related to Neural Networks

1. **S. Bazrafkan**, A. Kar, and C. Costache, "EYE GAZE FOR CONSUMER ELECTRONICS: CONTROLLING AND COMMANDING INTELLIGENT SYSTEMS", Consumer Electronics Magazine, IEEE, 2015, Vol: 4(4),pp 65 - 71
2. **S. Bazrafkan**, T. Nedelcu, C. Costache and P. Corcoran, "FINGER VEIN BIOMETRIC: SMARTPHONE FOOTPRINT PROTOTYPE WITH VEIN MAP EXTRACTION USING COMPUTATIONAL IMAGING TECHNIQUES," 2016 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2016, pp. 512-513.
3. A. Kar, **S. Bazrafkan**, C. Costache and P. Corcoran, "EYE-GAZE SYSTEMS; AN ANALYSIS OF ERROR SOURCES AND POTENTIAL ACCURACY IN CONSUMER ELECTRONICS USE CASES," 2016 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2016, pp. 319-320.

Chapter 2

Deep Neural Networks

The power of deep learning first made worldwide news in 2015, when a deep learning algorithm achieved better-than-human visual pattern recognition in an international competition. The accuracy was six-times better than the nearest nonneural network approach and twice as accurate as human experts [50]. After first interaction with DL, the impression is the simplicity of the technique [51]. Figure 2.1 shows the typical steps in producing a DNN. The network building blocks, loss functions and optimization techniques are explained in more details in sections 2.1, 2.3, and 2.4 respectively.

Considering the available toolkits and libraries, such as Tensorflow [13], Caffe [14], Theano [11], Lasagne [12], pytorch [15] and MXNet [16], the implementation of DL algorithms are quite convenient. However, after acquiring more experience with DNN models and dealing with different problems one would realize there are many details and pitfalls need to be taken care of. Data is the most important part of any data driven machine learning method including deep learning. Data augmentation is the process of expanding a database

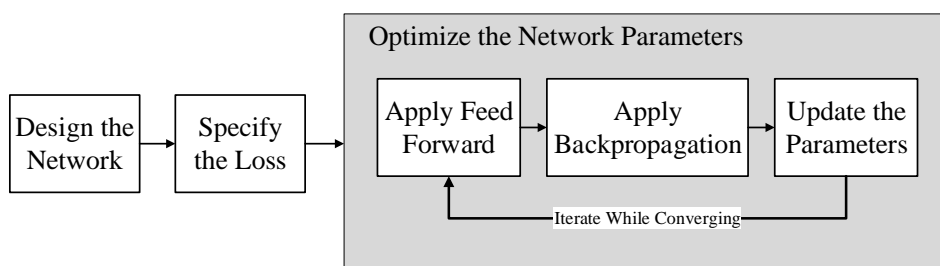


Fig. 2.1 The typical pipeline of model preparation in Deep Learning.

particularly for a specific application (see section 3.2.) There are questions which no absolute answers are provided in DL science such as:

- What is the most optimized network structure for a given dataset?
- What is the best data augmentation technique for a problem?
- What is the best optimization method in training neural networks?
- What is the best initialization for the network parameters?

Moreover, there are several other questions which are not trivial to answer due to the lack of general understanding of what is happening inside a DNN.

Information Bottleneck [52] is the method to determine the compression bound for a given dataset while preserving the mutual information with another set of samples. There are several attempts [53–56] of understanding the training procedure and information flow inside a DNN using the Information Bottleneck technique. In these approaches, the network is considered as a compression tool which forgets the unimportant information of its input in reconstructing the output data. The Information Bottleneck technique is applied to the model but unfortunately, there are no substantial outcomes on these observations and the data distribution evolution inside the network is not explained clearly.

In this chapter, the deep neural network's structure, training methods, and challenges are explained in more detail.

2.1 Neural Networks Building Blocks

Neural networks typically have an input layer, an output layer, and one or more so-called hidden layers (Figure 2.2). These layers are full of nodes often called neurons, which are connected to subsequent and previous layers using a number of schemes. Perhaps the most common connection scheme is the fully connected layer, in which every neuron in a layer is connected to every neuron in the previous and next layer. This idea is inspired from biological neurons, where we have axons and dendrites that connect individual neurons to each other. In the biological model, axons receive input from other neurons and dendrites transmit information to other cells [57]. This corresponds to the input and output connections in a neural network. The concept of multiple connection schemes also comes from biology, where we see unipolar, bipolar, multipolar, and pseudounipolar connection mechanisms.

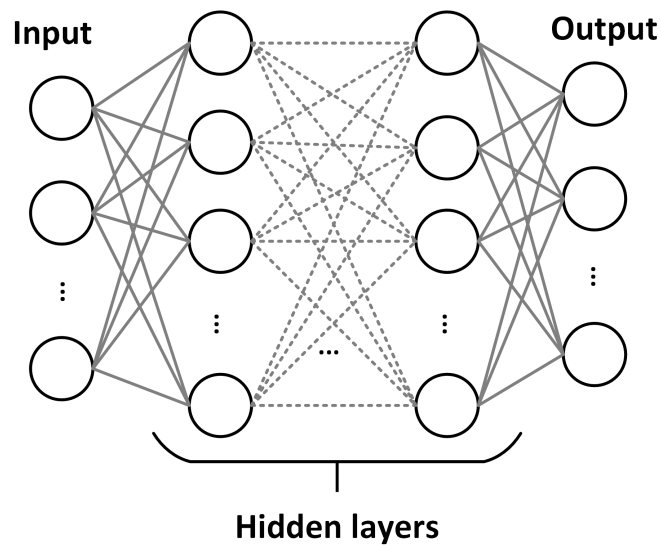


Fig. 2.2 A typical ANN with a fully connected hidden layers.

The body of a biological neuron is called the soma, which can decide when and what to transmit on the basis of various criteria. Artificial neurons have a similar mechanism called the activation function. This model of neurons (see Figure 2.3) was first invented in 1943 by Warren McCulloch and Walter Pitts [58]. The first popular implementation of these in computers was formalized in 1957 by Frank Rosenblatt in the idea of the Perceptron [59].

To store and process analog signals (e.g., voice and biological signals) on a computer, one must first convert these inputs into a digital form in a process called analog-to-digital conversion. This transforms the information from a continuous space to a discrete space. Some information is lost in the process. Often that information isn't critical to understanding the underlying analog signal, but in some instances, we could lose critical data, such as high-frequency information. In digital processing, we often refer to a piece of data, such as a picture, as a sample. It is convenient, but computationally expensive, to represent these samples in a high-dimensional space where each unit (or pixel, in the case of images), is considered as being located on a specific axis with the range of possible values being the size of that axis (e.g., 0–255 in the case of an 8-b color-channel in an image). An image that is 100×100 pixels would be represented as a vector that is a point in a 10,000-dimensional space. We call this space the feature space. Since even the most powerful computers can have trouble with such high-dimensional space, we try to reduce the number of dimensions to just those that are critical to a task or to change the way these features are represented. In the traditional pattern recognition approach, to perform a given task, we separate our process into

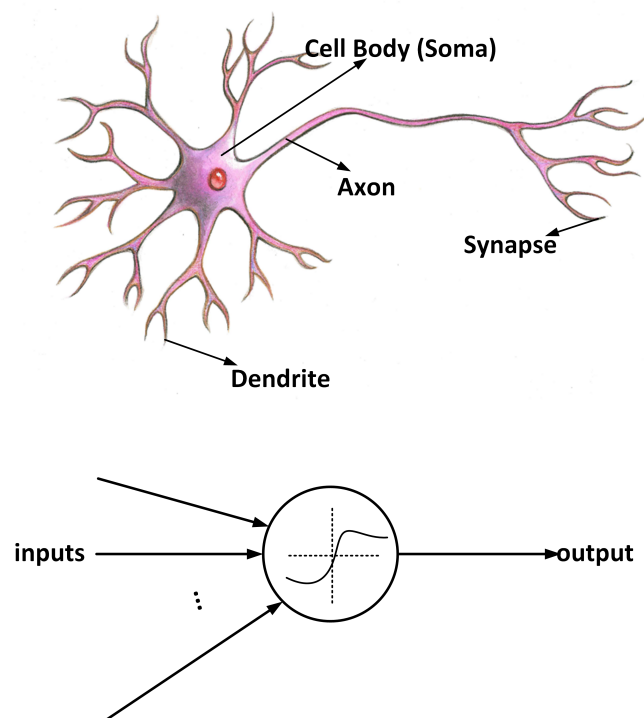


Fig. 2.3 (a) A biological neuron (illustrated by Kimberly Sowell, used with permission) and (b) an artificial neuron.

two steps: feature generation and feature selection. The former generates new features from the pixel space, while the latter reduces the dimensionality of the feature space. Examples of feature generation include morphological, Fourier, and wavelet transforms, which create more useful features for specific tasks. Feature selection includes methods like principal component analysis and linear Fisher discriminant [60]. A newer technique based on sparse mapping calls for expanding the dimensions with the goal of representing more abstract features instead of trying to reduce the dimensions. This is inspired by models of the visual cortex of animals [61].

Convolutional layers, a key component of deep learning, make use of this sparse mapping approach. One of the most novel and useful aspects of CNNs is that they can learn the filters that previously had to be custom-designed by the researcher, a task that would often take years of trial and error. Convolutional layers are essential to this task in modern deep neural networks. These layers make use of the convolution operator. The convolution operator is used on two functions. One is the signal from the sample space and the other, called the filter, is applied to the sample. On a GPU, convolutions are implemented as matrix multiplications. This operator has a long history in image processing applications and dates back to the time when digital image processing started. The convolution operation can be discussed in both spatial and transform space. In the spatial space, the convolution operator is the equivalent operation of correlation with the reversed filter, i.e., this operator calculates the similarity of the input function with the filter. For example, the edge detection and corner detection filters use the similarity of the input image with a predefined filter mimicking the edge or corner shape.

This is also happening in the transform space. Looking at the function in the Fourier space, the convolution is performing frequency filtering. For example, low-pass or high-pass filters have their equivalent spatial filters, which could be applied to the image using convolution operations (see Figure 2.4).

These filters used to be designed based on the observations and problem definition. Edge and Corner detector filters are designed based on the edge and corner models; and high and low pass filters follow the Fourier model of the signal. All these filters are designed based on the knowledge of the problem and they usually take a large amount of research and time to design. In the deep learning approach, the filter is learned and applied to the data during the training process with the hope that, after training, the learned filter will be the best choice for the task. One difference between the way convolutions are used in CNNs from more

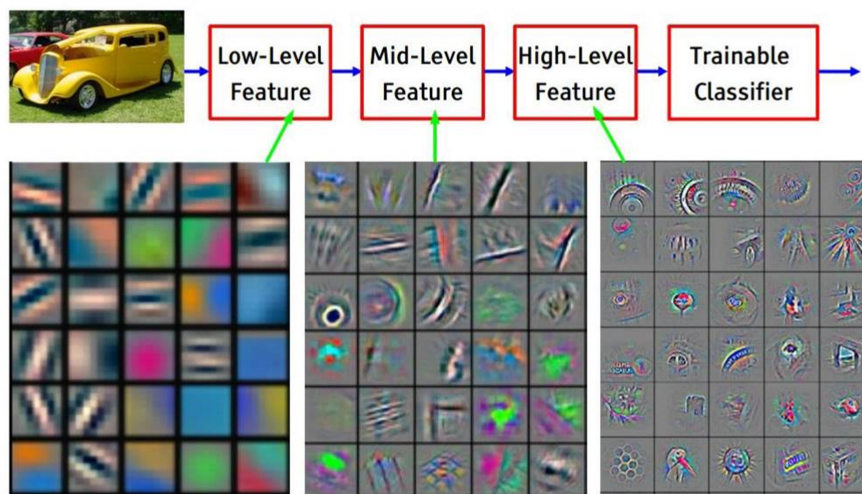


Fig. 2.4 Visualization of the learned convolutional filters at different layers. (Copyright Movidius, used with permission.)

traditional uses is that the convolution operator is applied using a four-dimensional (4-D) filter. This is essentially a set of three-dimensional (3-D) filters that are stacked in the fourth dimension. We use 4-D filter to map one 3-D space to another 3-D space (see Figure 2.5).

2.1.1 Convolutional Layer

In general, for an n -dimensional signal, the convolutional layer is an n or $(n + 1)$ dimensional feature space mapping with $n + 1$ or $n + 2$ dimensional kernels (filters). In the well-know case of a 2D image, the convolutional layer is 3-D with a 4-D kernel. In this case, the four dimensions of the kernel correspond to

1. the width of the input
2. the height of the input
3. the number of channels of the input
4. the number of channels of the output.

In Figure 2.5, you can see two different convolutional layers. The k channel layer on the left side is mapped to a p channel layer on the right side using a 4-D kernel. This kernel is shown using different 3-D kernels with different colors.

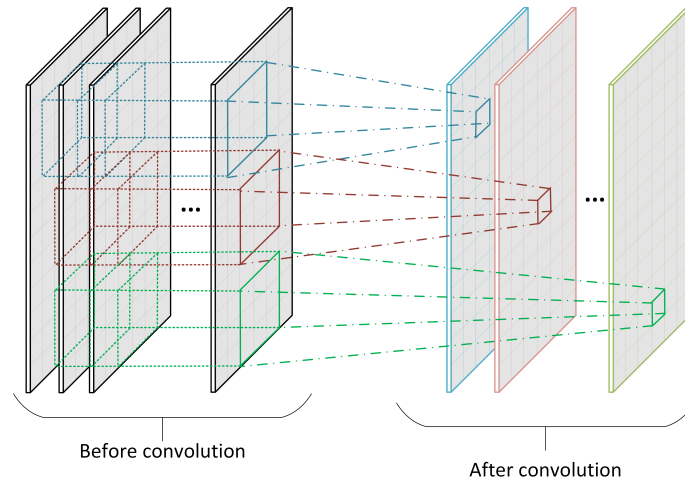


Fig. 2.5 (a) A biological neuron (A 4-D filter maps one 3-D space to another 3-D space using convolutions).

2.1.2 Pooling Layer

Pooling is an operation that accepts a pool of data values as input and generates one value from them to be passed to the next layer. This operation is usually mean or maximum of the input values. There are two important purposes for pooling operations. One is reducing the size of the data space to reduce overfitting, and the other is transition invariance. A pooling layer is performing the pooling operation on its inputs. Figure 2.6 shows how a pooling operation is applied to a one-channel input to reduce the dimensionality of the dataspace. In Figure 2.6, we have a 3×3 pooling operation applied to a 12×6 one-channel feature space. The most frequently used pooling operation is max-pooling, wherein the maximum value of the features in the pool is selected to be mapped to the next layer.

The importance of this layer is described in the next example. In generic deep neural networks, a dense, fully connected layer emerges from the convolutional layers. For example, consider a convolutional layer with ten channels and a 100×100 feature space. Placing a fully connected layer after it would result in computing 100,000 weights from this layer to each neuron in the dense layer, which requires significant memory and computation resources. Using a pooling layer helps reduce resource demands. Pooling also helps provide transition invariance by helping each kernel cover more space. Additionally, without a pooling layer, a network this big might suffer from overfitting, especially if there is not enough representative data in the training set. Overfitting is happening when the number of the learnable parameters

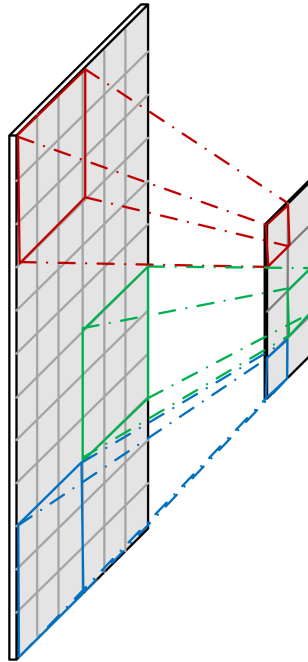


Fig. 2.6 A pooling operation reduces the size of the feature space.

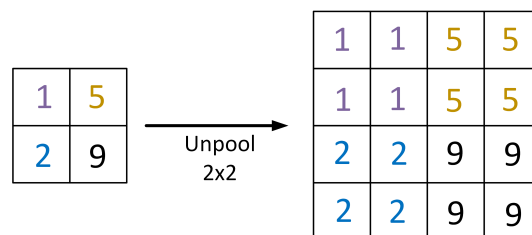


Fig. 2.7 A 2×2 unpooling operation with repeating values.

in the model is more than what the data can train. In this case the model remembers the samples in the training set while it can not perform generalization to other subsets of data.

2.1.3 Unpooling Layer

Unpooling layers are designed to perform the inverse operation of pooling layers by increasing the size of the feature space. These layers operate on a single pixel and expand that pixel into a pool of data. There are different implementations of unpooling layers.

The most popular are repeating values and sparse unpooling in deep neural network designs. The repeating value technique is expanding the data from one value to a pool of data with the same value (Figure 2.7). With sparse unpooling, the values of the original data

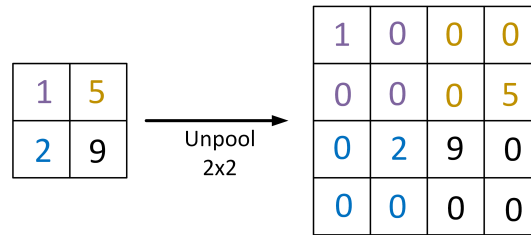


Fig. 2.8 A 2×2 sparse unpooling operation.

are mapped to a larger space in sparse form. Figure 2.8 illustrates 2×2 sparse unpooling. There are different methods for choosing the location where the value is mapped in sparse unpooling. For example, in [62] the indices have been memorized while applying the pooling layer, and in the unpooling layer, those indices are used to map the values.

2.1.4 Fully Connected Layers

Fully Connected Layers also known as Dense Layers (see figure 2.2) are exactly the same as classical neural network layers, where all of the neurons in a layer are connected to all of the neurons in the subsequent layer. The neurons give the summation of their input multiplied by their weights, which is then passed through their activation functions. Even more than with convolutional layers, these can cause the network size to grow, and so, typically, only one or two fully connected layers will be used in most deep networks.

2.2 Activation Functions

Each neuron in the deep neural network model is taking advantage of a nonlinear activation function to calculate the output value. This leads to one of the most advantageous properties of the deep neural networks: their ability to describe highly nonlinear systems. Being highly nonlinear helps the model be suitable for real-life problems and gives solutions in pattern recognition that cannot be achieved through more classical methods.

In the early days of the neural networks, the *tanh* and *sigmoid* activation functions were most popular. However, with the beginning of DNN which introduces numerous layers in a single model, more elaborated activation functions have been introduced to guarantee the convergence and simplicity of the network implementation. The most popular one is

Rectified Linear Unit (ReLU) [63], given by:

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \quad (2.1)$$

This function introduced with biological motivation and has several advantages over classical nonlinearities including but not limited to sparse activation, better gradient propagation and simplicity in implementation [64]. There are several variations of ReLU like LeakyReLU [65] and parametrized ReLU [66] which allow values in the negative range. Exponential Linear Unit (ELU) [67] is another variation of ReLU which improved learning characteristics compared to other activation functions including the speed of learning and generalization performance. ELU is defined by:

$$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \quad (2.2)$$

This function int There are numerous other modifications of ReLU and ELU like Randomized Leaky Rectified Linear Unit (RLReLU) [68], Scaled Exponential Linear Unit (SELU) [69] and S-shaped Rectified Linear Unit (SReLU) [70] each has their advantages and complications.

A Self-Gated Activation Function known as SWISH [71] is presented recently, and it has been shown that it works better than ReLU on deeper models. This activation function is described by:

$$f(x) = x \cdot \text{sigmoid}(x) \quad (2.3)$$

The authors in [71] claim that “While it is difficult to prove why one activation function outperforms another because of the many confounding factors that affect training, we believe that the properties of Swish being unbounded above, bounded below, non-monotonic, and smooth are all advantageous.”

There are other nonlinearities which are not a function of a single variable from the previous layer. For example, softmax [72] takes into account the output of all the other neurons while calculating a single output. This activation function for i 'th output is given by:

$$f_i(\mathbf{x}) = \frac{e^{x_i}}{\sum_{j=1}^N e^{x_j}} \quad \text{for } i = 1, 2, \dots, N \quad (2.4)$$

where N is the total number of outputs. This function is used in the last layer of classifiers because it represents the probability distribution over a set of outcomes and at the same time performs the normalization on the output layer which reduces the influence of extreme values and outliers.

Choosing the competent activation function is one of the critical challenges in DNN science. In [73] authors compare the impact of different nonlinearities on convergence and performance for several models. In this work several nonlinearities including RELU, LeakyReLU, parametrized ReLU, ELU, SELU and SWISH are compared in Inception-ResNet-v2 [74] and MobileNet [75] trained on ImageNet [76] and the results shows the superior accuracies while using SWISH.

2.3 Loss Functions

In machine learning, mathematical optimization, and decision theory a loss function is a mapping of the output of the model to a cost value which should be minimized using an optimization technique. In DL the loss function is also known as the objective function which generally describes a distance between the output of the network with the ground truth data. And the optimization is performed to the trainable parameters. For classification problems, the ground truth is also known as a label.

Selecting the proper loss function is of crucial importance in designing adequate model for a given problem. The most popular loss functions for classification problems are binary cross-entropy (for two class problems) and categorical cross-entropy (for multi-class problems.) The binary cross-entropy is given by

$$L_{bce} = -t \log(p) - (1 - t) \log(1 - p) \quad (2.5)$$

where t and p are the target and the prediction values respectively. and categorical cross-entropy is defined by

$$L_{cce} = - \sum_j t_{i,j} \log(p_{i,j}) \quad (2.6)$$

wherein i and j are the index of samples and network outputs respectively and t and p are the target and the prediction values respectively. There are other loss functions for classification purposes such as binary and multiclass hing losses [77].

When dealing with regression problems, the loss function became a matter of design. The most popular loss function for regression problems is Mean Squared Error (MSE) given by

$$L_{mse} = mean(\sum_i \sum_j (p_{i,j} - t_{i,j})^2) \quad (2.7)$$

where i and j are the index of samples and network outputs respectively and t and p are the target and the prediction values respectively.

Regressions loss is usually adjusted to the problem's behavior. For example in [78], the loss function consists of three parts: Appearance Matching Loss, Disparity Smoothness Loss, and Left-Right Disparity Consistency Loss. Selecting these specific cost functions assures the high quality of disparity reconstruction. There is another kind of loss function which is not applied to the samples directly. For example in [79] the Kullback-Leibler divergence between autoencoder's bottleneck and the ground truth is minimized. The objective is to increase the similarity between the distribution of the network's output and the target distribution. Based on the application, the goal justifies the distance increment as well. For example in [80], the objective is to find a latent representation of faces which is unique for each person. This is done by introducing a triplet loss which reduces the distance of the latent vector between samples of a particular individual while increasing the same distance between different persons. Ideally, this results in a network which maps different images of the same person into a particular vector while generates a different vector for a different individual.

Having the representative loss function plays a vital role in training a high-performance network which in most of the regression cases it is defined by the nature of the problem. In our works, a new regression loss function is introduced in the VAR+GAN project [49]. This function is designed to increase the distance between the distributions of samples generated for any two sets of aspects. This project is described in detail in Appendix I.

2.4 Optimization Algorithms

Optimizing the DNN parameters is the most sophisticated step in the training procedure. Most of the DL toolkits translate the network into a graph and calculate the derivatives of the loss function with respect to every single learnable parameter in the corresponding graph using the back-propagation method which utilizes the chain rule to compute the gradient in each layer.

Table 2.1 Essential optimization methods for DNN training.

Method	Advantage	$\Delta\theta_{(i)}$
SGD	Faster convergence than GD	$\eta\nabla_{\theta}F(\theta; \mathbf{X}^{(j,j+k)}, \mathbf{Y}^{(j,j+k)})$
SGD + momentum	Less fluctuations in the cost	$\gamma\Delta\theta_{n-1} + \eta\nabla_{\theta}F(\theta)$
SGD + Nesterov momentum	Captures the minimum more efficiently	$\gamma\Delta\theta_{n-1} + \eta\nabla_{\theta}F(\theta_{n-1} - \gamma\Delta\theta_{n-1})$
AdaGrad	Assigns learning rate to each parameter	$\frac{\eta}{\sqrt{G_{n-1,ii} + \epsilon}} \nabla_{\theta_i} F(\theta_i)$
AdaDelta	solves the learning rate vanishing problem of AdaGrad	$\frac{RMS[\Delta\theta]_{n-1}}{RMS[\nabla_{\theta}F(\theta)]_n} \nabla_{\theta} F(\theta)$
ADAM	Assigns a learning rate and momentum to each parameter	$\frac{\eta}{\sqrt{\hat{v}_n + \epsilon}} \hat{m}_n$

There are two main types of optimization methods used in ML. First-order and second-order methods [81]. The former one utilizes the first order derivatives of the loss function with respect to the learnable parameters to minimize the loss function, while the second order methods use the second order derivatives also known as Hessian matrix in optimizing the cost. The second order methods have the advantage of taking into account the curvature of the loss by using the information about the change in derivatives. The problem with the second order methods is their computational complexity especially when the second order derivatives are not known.

The first-order techniques use the gradient of the loss function with respect to the trainable parameters which is also known as the Jacobian matrix to minimize the cost function [6]. These methods are conveniently implementable and converge rapidly on big datasets [82].

The most popular first-order method is Gradient Descent (GD) optimization. Given by:

$$\theta_n = \theta_{n-1} - \eta\nabla_{\theta}F(\theta; \mathbf{X}, \mathbf{Y}) \quad (2.8)$$

wherein θ_n is the set of parameters in epoch n , F is the model's mapping function, η is the learning rate, and \mathbf{X} and \mathbf{Y} are the input data and targets respectively.

In this method, the weights and biases are moved in the opposite direction of the gradient which guarantees the decrease in the cost value. DL techniques use different variations of GD to ensure better performance and faster convergence. In the original GD method, the gradient is calculated after passing all the samples through the model and the parameter update after each epoch. This causes very slow convergence, especially for big datasets. Mini-Batch Gradient Descent also known as Stochastic Gradient Descent (SGD) solves this problem by processing a batch of samples at a time. It has the advantage of reducing the variance of the update which stabilizes the convergence and also introduces the possibility of getting more out of highly optimized deep learning software toolkits which perform matrix multiplications on GPUs [83]. One can accelerate the convergence by applying a momentum towards the minimum value [84]. The main issue with the momentum method is when the algorithm gets close to a minimum. The momentum value pushes the parameters to jump over the optimal point and continue to go uphill. The Nesterov Momentum method [85] solves this problem by reducing the momentum value when getting close to a minimum point. In other words, it slows down the speed around optimum values.

One of the most significant difficulties with any gradient-based method is to find the proper learning rate, and also how to manipulate it during the training procedure. AdaGrad [86] solves this problem by assigning learning rate to each parameter in a way that more frequent parameters get smaller learning rate and vice versa. This method induced a problem known as vanishing learning rate which causes slow convergence compare to other gradient-based techniques. The AdaDelta method [87, 88] solves this problem by restricting the memory for the gradient to a finite number of previous steps.

In AdaDelta method, the learning rate is adjusted for each parameter, but all parameters share the same momentum value. The ADAM [89] technique improves this matter by assigning a learning rate and a momentum for each learnable parameter. This optimization method presents a solution to problems such as vanishing learning rate, slow convergence, and fluctuations in the cost value and is one of the most popular optimization techniques which is implemented in most of the DL toolkits.

Table 2.1 shows the important optimization methods used in deep neural network back-propagation. In our works, we widely used Nesterov momentum for its power in capturing minima and fewer fluctuations; and also ADAM optimizer because of its ability to assign different learning rate and momentum to each parameter which makes it a reliable method in training procedures.

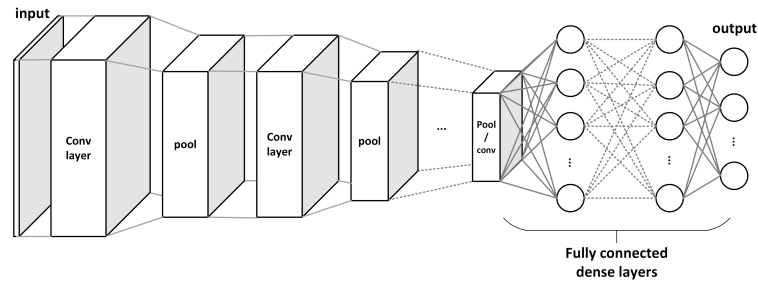


Fig. 2.9 A generic deep neural network, with convolutional (conv) and pooling layers followed by fully connected dense layers.

2.5 Deep Neural Network Architectures

Designing the best neural network for a given problem is still an open issue in the DL science. But knowing the structure of the input and output data provides a solid starting point for the design procedure. In this section, the most commonly used architectures for DNNs are presented.

1. **Generic deep neural networks** are usually made of one or more convolutional layers, wherein each convolutional layer is generally accompanied by a pooling (max-pooling) operation. One can also use bigger strides in the convolution layer to reduce the data dimensionality (the stride of a convolution operation is the number of the pixels the kernel window is sliding before calculating the convolution in each location). In generic deep neural networks, the convolutional layers are usually followed by one or more dense fully connected layers (Figure 2.9). The rectified linear unit is the most common activation function used in these networks. The last layer is typically taking advantage of other nonlinearities based on the task. In our works, this type of networks are used in facial expression classification [46] (Appendix C), as the auxiliary classifier and regressor in VAC+GAN [47, 48] (Appendices G and, H) and VAR+GAN [49] (Appendix I) projects and as classifier network in the Smart Augmentation [3] (Appendix E) project.
2. **Fully convolutional Networks** are deep neural networks in which all of the layers are convolutional, pooling, and unpooling layers, wherein the pooling and unpooling layers are usually placed between two convolutional layers. They are similar to typical deep neural networks except they have no dense layer. The output of a fully convolutional network is as the same type as its input. For example, if the input is a k -channel

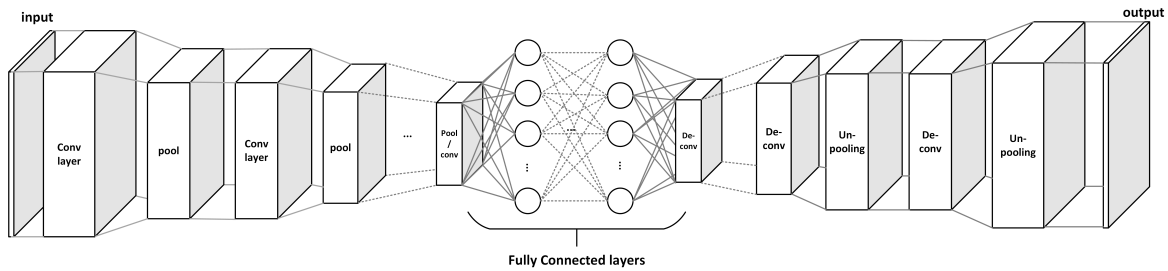


Fig. 2.10 An autoencoder is the merged structure of the encoder and decoder in a model.

image, the output of the network could be a p-channel image but not something else entirely. We used a fully convolutional network in the iris segmentation project [2] (Appendix D) where the input is the low-quality eye image and the output is the iris segmentation map, and also the augmentor networks in the Smart Augmentation project [3] (Appendix E) are fully convolutional.

3. **Autoencoders (AE)** are a deep neural network design in which the input and output data are from the same class and also have the same data structure. For example, if the input of the network is a three-channel, 128×128 image, the output of the autoencoder is also a three-channel image with the size 128×128 . The autoencoder network could be a fully convolutional network, or it can have one or several fully connected layers in the middle of the network, which is usually known as the bottleneck of the network. The idea with this kind of network is to create a compressed version of the input in the bottleneck. The data can then be reconstructed from the bottleneck. The part of the network that does the compression is called the encoder. The part that decompresses the data from the bottleneck is called the decoder. The autoencoder refers to the merged structure of the encoder and decoder in a model (Figure 2.10). AEs play important roles in different applications, for example the BEGAN implementation uses an Autoencoder network as the discriminator part. We used BEGAN in VAC+GAN [47, 48] (Appendices G and, H), VAR+GAN [49] (Appendix I), and GAN+aspect (Appendix F) projects.
4. **Fully Connected Networks** These networks which are also known as Multi-Layer Perceptron (MLP) networks are the oldest implementation of neural networks. In these architectures, all the neurons in a layer are connected to every neuron in the next layer. An MLP with more than one hidden layer is considered as a deep network. In our work, fully connected networks have been used in the GAN+aspect project (Appendix F) to map the latent space into the aspect space.

Chapter 3

Contributions and Methodologies

In late 2015 with the fast emerging of Deep Learning, we decided to turn into using this technique instead of the classical machine learning methods in our research. This was happening while the DL was not mentioned in the original framework of my Ph.D. course. In the time with the inevitable influence of DL in every aspect of AI this was a wise decision to take.

We started with applying an inter-class evaluation to investigate the real world performance of a DNN for facial expression classification task. This project is explained in detail in [46] (Appendix C, Section 3.2).

Next project was to use Fully Convolutional DNN to segment the low-quality iris images for wild use cases. The idea of Semi Parallel Deep Neural Networks (SPDNN) originally started from this project, when suddenly we realized that by mixing and merging several networks and remove redundancy of the repeated layers we can get superior results even while keeping the number of parameters relatively same to the original networks. This idea is described in detail in [2, 45], (Appendices A and D, Section 3.1). The monocular depth estimation project [1] (Appendix B) took advantage of the SPDNN method as well.

Neural network design was not the only contribution to the iris segmentation project. A great lesson learned from this project was how much data augmentation is important in getting superior results. This way of thinking led us to introduce a new methodology known as Smart Augmentation wherein several photos from a specific class are merged nonlinearly to create a new sample from the same class. This method is explained in [3] (Appendix E, Section 3.2.2).

Augmentation techniques play a crucial role in expanding datasets for data-driven techniques especially with the introduction of new regulations on the privacy and data security such as GDPR which turns the data acquisition into engineers nightmare. The Generative Adversarial Networks (GAN) are one of the best candidates to learn the existing database distributions and draw randomly generated samples from the learned distribution. We worked on this technique to learn the joint distribution of data and ground truth (Appendix F, Section 3.3.1) and also expanded the idea of conditional generators which has appealing applications such as generating gender-specific samples (for more details see [47–49](Appendices G, H, and I, Sections 3.3.2 and 3.3.3)).

In the next section, the SPDNN technique and the related projects are explained followed by our contribution to data preparation and smart augmentation in the following section. And the last section is dedicated to the contributions on Deep Neural Generators.

3.1 Semi Parallel Deep Neural Networks

As previously commented, there has been a spectacular growth in research on artificial intelligence in general and on the application of deep networking techniques in particular. In each DNN design there are several factors need to be taken care of including the number of layers, layer types, nonlinearities, loss function and optimization method. Designing and training a DNN is not a trivial task especially when dealing with Big Data since the data distribution is not accessible. Most of the designs in the literature are taken from well-known models such as VGG [23], Inception [25], and Alexnet [19]. These networks are designed and trained for a specific task and borrowing their architecture for any other problem is not always a wise decision. The other difficulty arises when there are several network architectures designed for a specific problem, some give higher accuracy, and others have less computational cost. So much so, that faced with almost any contemporary machine learning problem, it is almost inevitable that one can find a multiple of varying network architectures derived from a number of core datasets in the literature. Typically each dataset is developed and annotated for this specific class of problem, and each DDN derived from these datasets has its own distinct advantages and drawbacks.

The challenge for engineers is to find the best network matched to their specific problem or design goal. But in practical ‘wild’ use-cases there is often no winning network and much of the recent literature improves on performance aspects by adding layers to deepen the network. While this may improve specific aspects of network performance, a price is paid

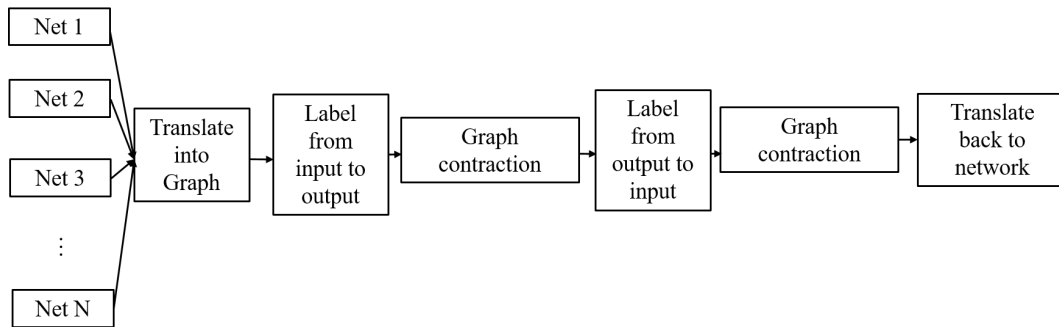


Fig. 3.1 SPDNN workflow. This technique merges several networks using graph contraction method.

in terms of additional memory requirements to store the network and additional compute cycles to process the added layers or greater areas of silicon if the goal is to provide a chipset implementation. And if the design engineer wishes to combine several of these deep networks in parallel to have the benefits of each, then additional logic or computation needed to merge the network outputs and the resource requirements can become unrealistic.

It is, naturally, possible to go back to the drawing board and design a completely new network from scratch, but designing, implementing, testing and optimizing deep networks is challenging and time-consuming, as is the creation of new annotated datasets to enhance and focus the network capabilities. The Semi Parallel Deep Neural Networks (SPDNN) proposes a solution to this issue by introducing a method to mix and merge several DNNs into a single model. In this technique, each network translates into a graph and after assigning proper labels to each node, the graph contraction method merges all networks into a single model. See figure 3.1 This technique is explained step by step as follows:

1. Translate the network into a graph and perform labeling from input to output.
2. Apply graph contraction on the graph.
3. Apply the labeling from output to input.
4. Apply graph contraction.
5. Turn back the graph into the neural network structure.

In order to turn a neural network into a graph:

1. Consider each layer of the network as a node of the graph.

2. Connect each node to other nodes based on their connection in the original network. If two layers A and B are connected in the network. Two nodes A and B will be connected in the corresponding graph.
3. Label each node based on the properties of each layer. Labeling is described below:
 - (a) The first property of each layer is its structure. For a convolutional layer the kernel size, for pooling/un-pooling layer the pool size and for the fully connected layer the number of the neurons.
 - (b) The second property of each layer is its distance from the input in the graph.
 - (c) We show the properties of each node as a tuple. (C stands for convolutional F for fully connected P for pooling and U for un-pooling, for example, 3C mean a convolutional layer with kernel size 3×3 , 4F means a fully connected layer with 4 neurons, and 2P mean a 2×2 pooling layer). The first term of the tuple is the structure of the layer and the second term is the distance from the input. For example, the tuple (5C , 3) means a convolutional layer with a 5×5 kernel which has distance 3 from the input node.
 - (d) Sticky and non-sticky properties: The structure properties “pooling” and “un-pooling” are sticky with respect to convolution. It means that if pooling is applied to a layer, the pooling property will stay with the data in the following convolutional layers. except when the graph reaches an un-pooling or a fully connected layer. Each un-pooling layer will remove one pooling layer and the fully connected layer will remove all pooling properties from the node. The structure properties “convolutional” and “fully connected” are non-sticky.
 - (e) If both properties for several layers are equal, the same label will be assigned to all of them.

Here, the SPDNN methodology is explained by an example:

Consider three different networks designed to perform a binary classification task shown in figure 3.2. Graph corresponding to each network is shown in the right side of this figure. Each node in the graphs stands for one layer. The properties for each node is written on the top of the node in the tuple (layer structure, distance from input).

The graph contraction task is a reasonably easy task in order to merge different vertices which leads to a simpler graph with less number of nodes. The task is given below

1. Merge all the input nodes to one input.

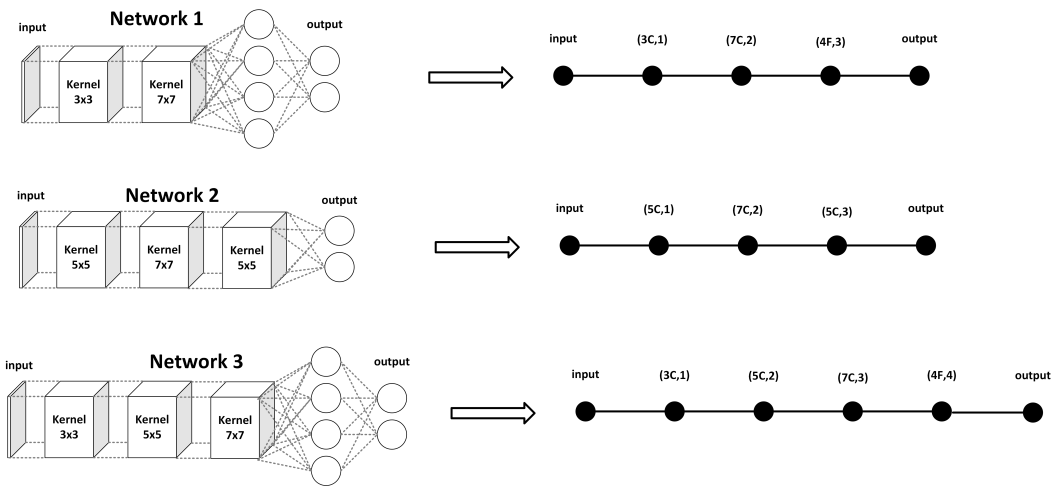


Fig. 3.2 Left: three different networks designed for a specific binary classification task. Right: the graph correspond to each network. The properties for each node is written on the top of the nodes in the form (layer structure, distance from input).

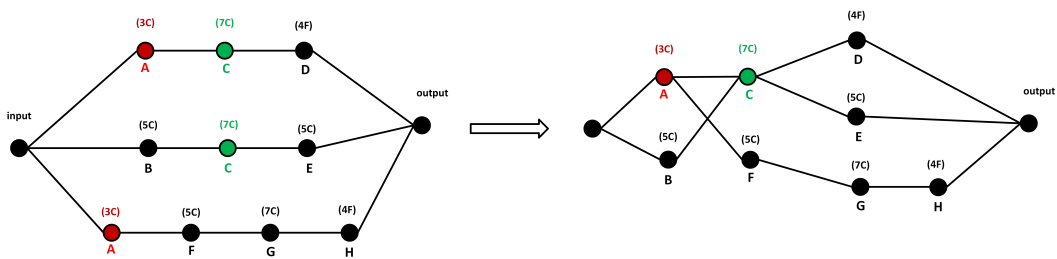


Fig. 3.3 Left: merge input nodes and out nodes in to single input and output. Right: apply the graph contraction to the graph on the left by merging nodes with the same label and removing the parallel vertices.

2. Merge all the output nodes to one output.
3. Merge the nodes with the same label.
4. Remove the parallel edges (if exists).

the graph contraction result for the graph shown in figure 3.2 is illustrated in figure 3.3.

The next step is to perform labeling from output to input. The properties of each node will be of the form: (layer structure, smallest distance from output node). The layer structure is still the same as the previous step. The nodes with the same property values get the same

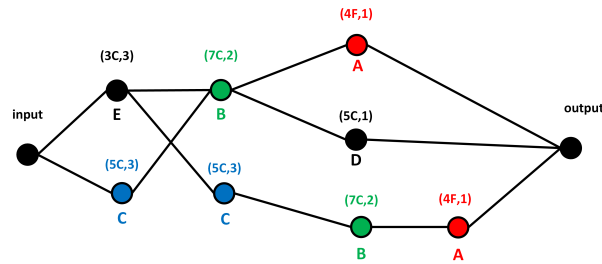


Fig. 3.4 Labelling for the graph in figure 3.3 (right). The tuples on the top of each node is (layer structure, distance to output node). The nodes with the same properties will get the same label.

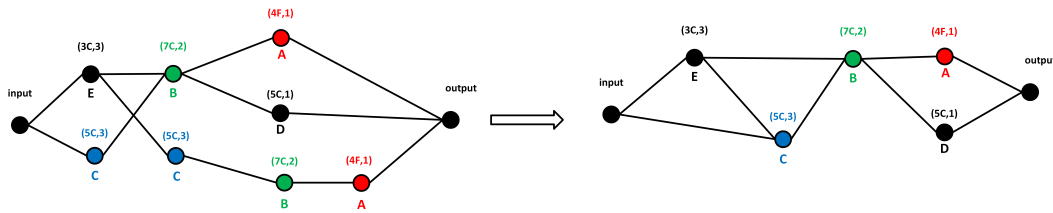


Fig. 3.5 The graph contraction is applied to the graph shown in figure 3.4.

label. Figure 3.4 shows the labeling from output to input.

The graph contraction task is the same as the previous graph contraction task given below:

1. Merge the nodes with the same label.
2. Remove the parallel edges (if exists).

Figure 3.5 shows the graph after applying the graph contraction for the second time. Since the layer structure is known from the node properties, the graph could be turned back into neural network structure. If two nodes are connected in the graph then those two layers will be connected to each other. The network correspond to the graph in the figure 3.5 is shown in figure 3.6.

In [45, 1, 2] (Appendices A, B and, D) several examples of SPDNN applications are presented on iris segmentation and depth estimation problems. In [2] (Appendix D) a fully convolutional DNN has been designed by merging 4 deep neural networks using SPDNN to perform iris segmentation task in unconstrained scenarios. This work is described in Section 3.2.1

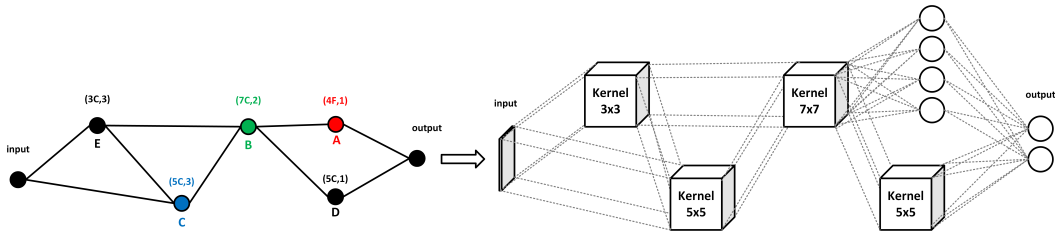


Fig. 3.6 The graph can be turned back to the neural network using the layer structure and the connections.

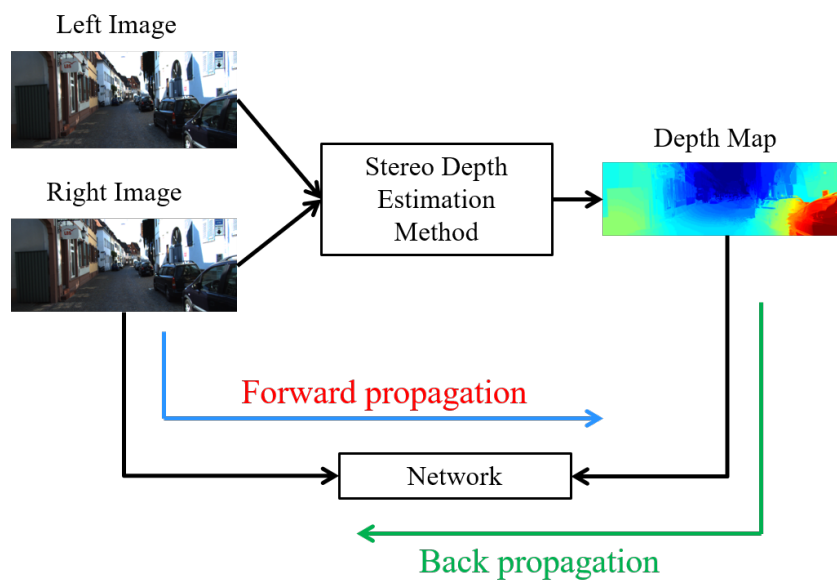


Fig. 3.7 Training workflow for work presented in [1] (Appendix B).

in [1] (Appendix B¹) 8 networks are merged into a single model to perform the depth estimation from a monocular camera setup. Four SPDNN models are trained and have been evaluated at two stages on the KITTI dataset. The ground truth images in the first part of the experiment are provided by the benchmark, and for the second part, the ground truth images are the depth map results from applying a state-of-the-art stereo matching method. The training workflow is shown in figure 3.7. The results of this evaluation demonstrate that using postprocessing techniques to refine the target of the network increases the accuracy of depth estimation on individual mono images. The impact of segmentation on depth estimation has been investigated in this work as well and it shows that the segmentation induces marginal improvements in depth estimation using monocular setups. In this work, there was two

¹Code available at: <https://github.com/shababqcd/SPDNN-Depth>

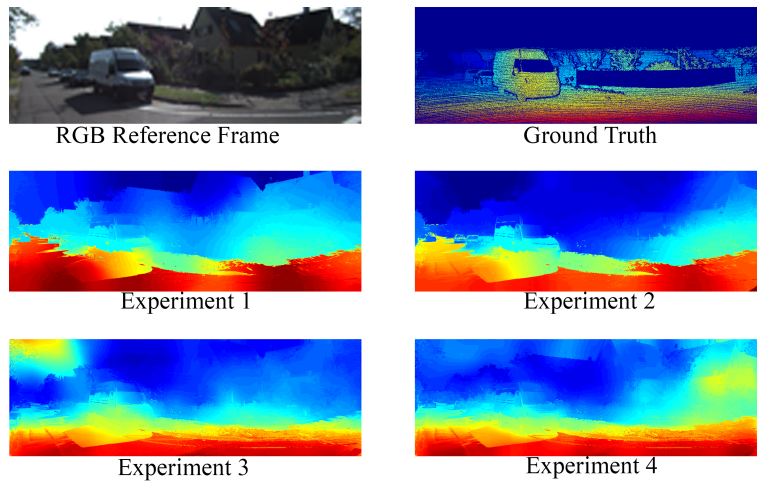


Fig. 3.8 Output of four experiments for a road scene.

sets of input (with and without segmentation) and two sets of targets (before and after post processing). Therefore there are four experiments conducted by training four networks. Output of these networks are shown in figure 3.8 for a road scene.

The SPDNN method has been employed in several other application including hand segmentation [90], and Denoising/Demosaicing problems.

3.2 Data Preparation

A big issue in the amateur deep learning community is the tremendous amount of concentration on the "network" compared to the "data." In this way of thinking, most of the credit is given to the "network"; and "data" is considered as an auxiliary part which helps the network to learn. In the researches which have been conducted in this thesis, the biggest lesson was to realize that the network should serve the data not the other way. In fact, the training data determines how big the network should be, what kind of layers should be used, what nonlinearity will serve better and what loss function should be employed in the model. Although finding the best model properties from the data itself is a nontrivial task and still an open problem to solve. One can expand and reorient the dataset to include more information and also represent more realistic scenarios. Being able to provide the most representative data for a given problem is a game-changing factor especially in real-world scenarios.

To improve the generalization of a neural network model, the most obvious solution is to increase the size of the dataset by providing new samples. This is investigated in [46]

(Appendix C) wherein different networks are trained on different datasets for the facial expression problem. Each network is tested on all databases to perform inter-database evaluations. A separate network is trained on the mixture of all databases, and again the same test procedure is performed. In this work, it has been shown that mixing and merging several databases has a productive effect on the generalization of the network output.

But acquiring new samples for a specific problem is not always easy since providing labeled databases needs significant human resources and is time-consuming. The other solution is to generate new samples by applying different functions and transformations to current ones. These operations are known as data augmentation. The use of augmentation in deep learning is ubiquitous, and when dealing with images, often includes the application of rotation, translation, blurring and other modifications to existing images that allow a network to better generalize [91]. Augmentation serves as a type of regularization, reducing the chance of overfitting by extracting more general information from the database and passing it to the network.

Many deep learning frameworks can generate augmented data. For example, Keras [92] has a built-in method to randomly flip, rotate, and scale images during training but not all of these methods will improve performance and should not be used “blindly”. For example, on MNIST (The famous handwritten number dataset), if one adds flipping, the network will be unable to distinguish properly between handwritten “6” and “9” digits. Likewise, a system that uses deep learning to classify or interpret road signs may become incapable of discerning left and right arrows if the training set was augmented by the indiscriminate flipping of images.

Applying any augmentation needs to be done based on the problem properties and the knowledge of the environment of the final solution. For example, if the network is trained for scenarios where a Gaussian Noise is present, adding this kind of noise to the training set turns the network robust to this kind of alteration.

3.2.1 Augmentation From an Expert Knowledge

In [2](Appendix D²) an iris segmentation solution is introduced for unconstrained scenarios. Since there are no big databases available for low-quality iris images, an augmentation technique has been deployed to turn high-quality images into wild, real-life samples. The

²Code available at: <https://github.com/shababqcd/IrisSegDNN>

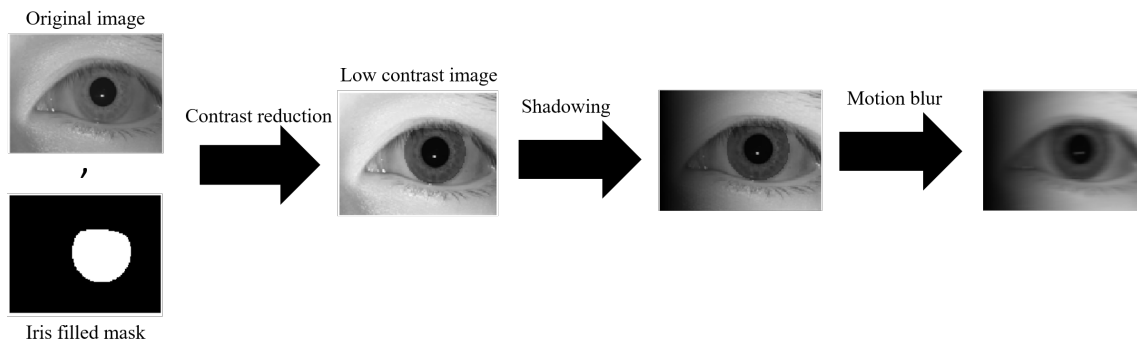


Fig. 3.9 Augmentation workflow applied in [2] (Appendix D).

difference between a high quality constrained iris images and consumer grade images depend on five different independent factors: 1- eye socket resolution, 2- image contrast, 3- shadows in the image, 4- image blurring, 5- noise level [93], [94]. Noise is a well-studied phenomenon and image de-noising can be done outside the network and also note that introducing high-frequency noise into the dataset trains a low-pass filter inside the network; apply de-noising outside the network gives a higher chance to use the whole network potential to perform the segmentation task. In addition, introducing Gaussian noise into the dataset will cause underfitting as explained in [95]. Therefore this work focuses on the first four factors in augmenting the iris database. Figure 3.9 illustrates the augmentation workflow. This augmentation technique alongside the power of the SPDNN method gives the best segmentation results compared to the state of the art techniques. The segmentation map generated by the network for some low equality iris images are shown in figures 3.10 and 3.11.

3.2.2 Smart Augmentation

Besides intuition and experience, there is no universal method that can determine if any specific augmentation strategy will improve results until after training. Since training deep neural nets is a time-consuming process, this means only a limited number of augmentation strategies will likely be attempted before deployment of a model.

Blending several samples in the dataset in order to highlight their mutual information is not a trivial task in practice. Which samples should be mixed together how many of them and how they mixed is a big problem in data augmentation using blending techniques. Augmentation is typically performed by trial and error, and the types of augmentation performed are limited to the imagination, time, and experience of the researcher. Often, the choice of

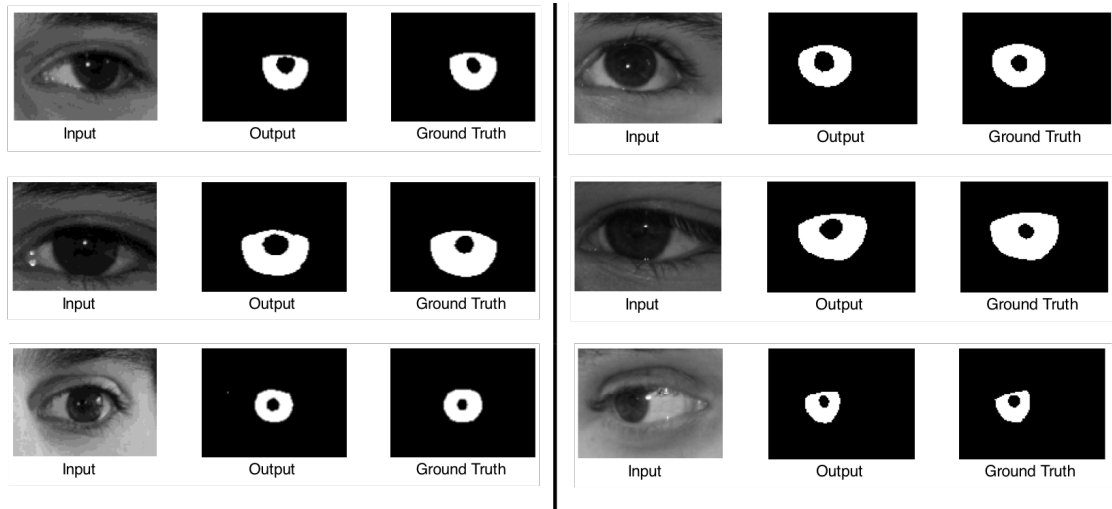


Fig. 3.10 Segmentation map generated by the network for low quality iris images.

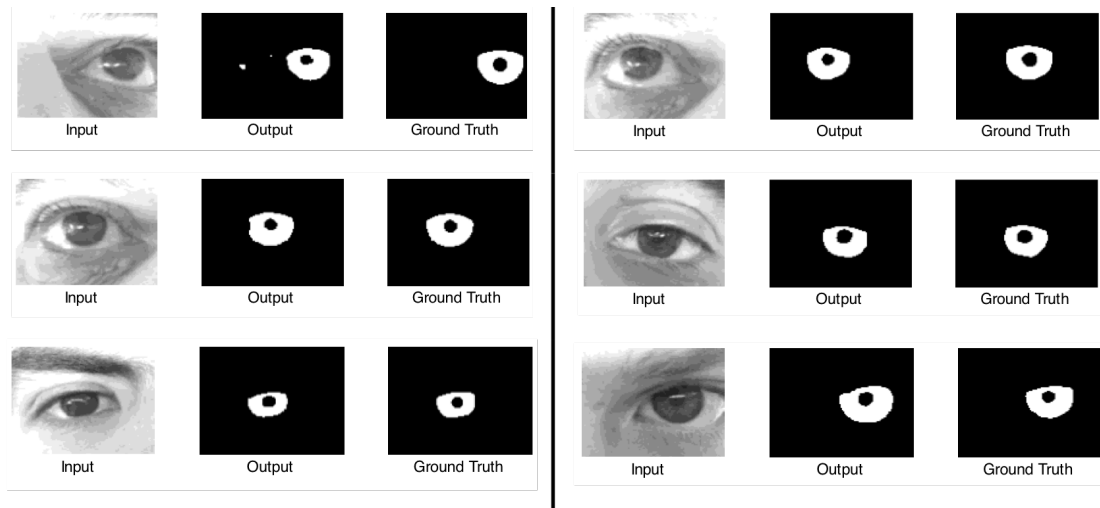


Fig. 3.11 Segmentation map generated by the network for low quality iris images.

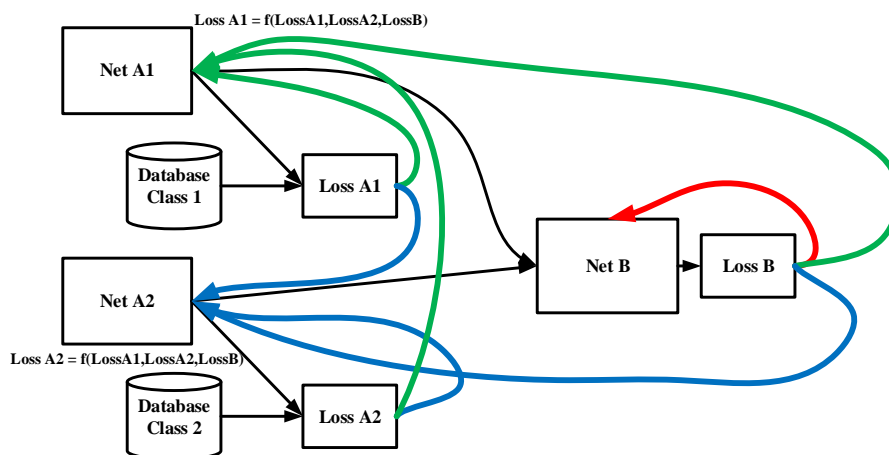


Fig. 3.12 Smart Augmentation technique explained in [3] (Appendix E). Black lines correspond to the forward propagation. Red line is backpropagation for NetB, green lines are backpropagation for NetA1 and blue lines are backpropagation for NetA2

augmentation strategy can be more important than the type of network architecture used [96].

Before Convolutional Neural Networks (CNN) became the norm for computer vision research, features were “handcrafted”. Handcrafting features went out of style after it was shown that Convolutional Neural Networks could learn the best features for a given task. In [3] (Appendix E), We suggest that since the CNN can generate the best features for some specific pattern recognition tasks, it might be able to give the best feature space in order to merge several samples in a specific class and generate a new sample with the same label. Our idea is to generate the merged data in a way that produces the best results for a specific target network through the intelligent blending of features between 2 or more samples.

The goal of Smart Augmentation is to learn the best augmentation strategy for a given class of input data. It does this by learning to merge two or more samples in one class. This merged sample is then used to train a target network. The loss of the target network is used to inform the augmenter at the same time. This has the result of generating more data for use by the target network. This process often includes letting the network come up with unusual or unexpected but highly performant augmentation strategies.

First, several samples from the training data are drawn from a class and fed to the augmented network called network A in this case. Based on the implementation, there could be more than one augmentor, each specified for a class. The augmentor's job is to merge its input images and produce an output which has the same properties as the input class. One loss is defined at the output of the augmentor known as augmentation loss. At the next step, the augmented data is fed to a classifier (also known as network B) alongside with the original samples. The classifier is updated based on the classification loss. But the loss function for the augmentor is a mixture of augmentation loss and classification loss. In this way, the augmentor learns to produce samples that decrease the classification error. The Smart Augmentation framework is shown in figure 3.12

The smart augmentation is an ongoing group project. My main contribution was preparing the script for train and test purposes. The biggest issue was in the implementation stage. The reason is the existence of several data paths in the network graph while each network is tied to its specific loss function. More than 30 observations including using several augmentors and trying different databases show the effectiveness of the Smart Augmentation method. The results include up to 6.7% increase in accuracy on AR Faces dataset, 6.1% on Audiance dataset and, 5.0% on FERET dataset compared to non augmented data. We also show that this methodology decreases overfitting, increases generalization capability and can significantly reduce the number of parameters required to perform the same task (i.e. a smaller network can work just as well as a large network)

Through several evaluations on different problem sets, the effectiveness of Smart Augmentation has been shown for classification problems. Figure 3.13 shows an example of how the augmentation network learns to merge two male images and generate a new sample from the same class.

3.3 Deep Generative Models

Deep neural networks have shown great success when used as generative models. These models learn the distribution of a specific dataset and can generate new samples from the learned Probability Distribution Function (PDF). There are classical methods include Variational Bayesian models, and Markov Chain Monte Carlo, which has been used to model the data distribution. Taking advantage of the neural networks non-linearity in defining the generative model goes back to early 2000 when [97] used *tanh*, the nonlinearity of a small neural network, in modeling the data distribution. These models are usually limited to

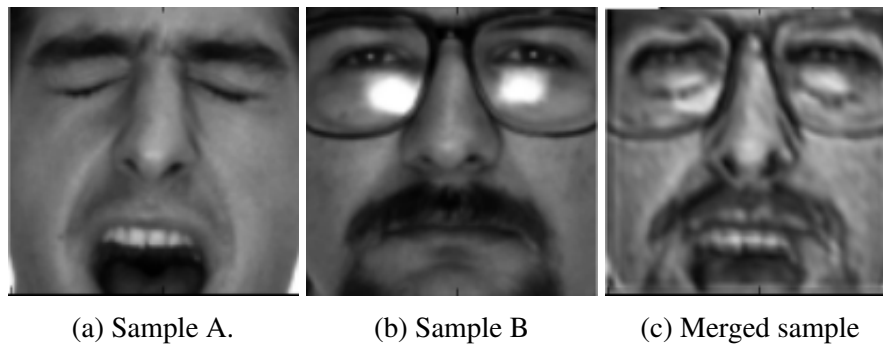


Fig. 3.13 Augmentation network merges several samples from a class to generate a new sample from the same class.

small-sized problems due to the complexity, and they are also computationally prohibitive.

With the emergence of the low-cost, high-performance hardware for training deep neural networks, it became feasible to train and test big networks, and recently the generative models have also been taking advantage of deep neural networks in learning large size problems including image and sound generation.

Generative Adversarial Networks (GAN) [38] utilise Deep Neural Network capabilities and are able to estimate the data distribution for large size problems. These models comprise two networks, a generator, and a discriminator. The generator makes random samples from a latent space, and the discriminator determines whether the sample is adversarial, made by the generator, or is genuine image coming from the dataset. GANs are successful implementations of deep generative models, and there are multiple variations such as WGAN [39], EBGAN [40], BEGAN [41], ACGAN [42], and DCGAN [43], which have evolved from the original GAN by altering the loss function and/or the network architecture. Variational Autoencoders (VAE) [79] are the other successful implementation of deep generative models. In these models the bottleneck of a conventional autoencoder is considered as the latent space of the generator, i.e., the samples are fed to an autoencoder, and besides the conventional autoencoders loss function, the Kullback-Leibler (KL) divergence between the distribution of the data at the bottleneck is minimized compared to a Gaussian distribution. In practice, this is achieved by adding the KL divergence term to the means square error of the autoencoder network. The biggest downside to VAE models is their blurry outputs due to the mean square error loss [98]. PixelRNN and PixelCNN [99] are other famous implementations of the deep neural generative models. PixelRNN is made of 2-dimensional LSTM units, and in PixelCNN, a Deep Convolutional Neural Network is utilized to estimate the distribution of the data.

In the following sections, contributions related to GAN are explained. In the most of these works, the Boundary Equilibrium Generative Adversarial Networks (BEGAN) [41] has been used to learn data distribution. One of the most challenging works in training GANs is getting them to converge to the desired distribution. In BEGAN, the discriminator network is an autoencoder and the generator architecture is the same as the decoder part of the discriminator. And the loss function is inspired by control theory, as follows: Suppose that x is the real data coming from the database, z is a sample from the uniformly distributed random space Z , D is the auto-encoder function with the loss defined by:

$$L(v) = |v - D(v)|^2, \quad (3.1)$$

where v is the input to the auto-encoder. The objectives for BEGAN are:

$$\begin{aligned} L_d &= L(x) - k_t \cdot L(G(z_d)) \\ L_g &= L(G(z_g)) \\ k_{t+1} &= k_t + \lambda_k (\gamma L(x) - L(G(z_g))) \end{aligned} \quad (3.2)$$

where L_D is the discriminator loss, L_G is the generator loss, $G(z)$ is the output of the generator for input vector z , k_t is the equilibrium hyperparameter, and λ_k is the learning rate for k .

In order to be able to use this implementation of GAN in other purposes, we needed to be able to reproduce the original results given in [41]. This was a challenging task since the code shared by the authors did not converge to the desired distribution. Therefore we reimplement the method ourselves and after several failed attempts, the network finally converged.

3.3.1 Latent Space Mapping for Generation of Object Elements with Corresponding Data Annotation

Being able to learn the data distribution and generate random samples is a powerful tool in the database expansion and augmentation process. But in order for the generated data to be useful in any further learning processes, one needs to generate the ground truth alongside the data itself. In this section, a new method is introduced wherein for a specific data generator, the corresponding aspect generator is trained. This method gives the opportunity to expand a dataset and also find the intermediate samples while generating the ground truth for any of those samples.

For a given database, if there is a perfect generator mapping a latent space into the data space (since the generator is a local operation on the latent variable) by considering the data processing inequality, one can argue that the latent space includes all the information of the database. This indicates that any aspect of the dataset is extractable from the latent space alone. The problem considered in this work is to find a local operation that maps the latent space to the aspect space for a given generator and aspect.

The generator presented by the GAN frameworks maps a uniformly distributed latent space into the data space. Note that, inverse transform sampling theorem declares that by knowing the probability distribution of a random variable x_r , there is a transformation, mapping a uniformly distributed random variable z_r into the space of x_r . Since the generator network accepts a uniform random variable and transforms it into the image space, this network is learning an approximation of the data distribution. At the same time, the aspects of the dataset are extractable using local operations on the data itself. Therefore the data processing inequality justifies the ability of a neural network to learn the distribution of any aspect of the data for a given generator network.

In Appendix F a method for learning the data aspect from the latent space is proposed. The training step of this approach is illustrated in figure 3.14

This method applies to any reversible generative model. For a trained generator the first step is to extract the latent space equivalent for each sample in the database. This step is also known as learning the latent space. Next step is to train a second network mapping the learned latent space into the aspect space. We showed that the simple Mean Square Error loss function is sufficient for training the aspect generator.

The inference step is shown in figure 3.15, where the latent variable is fed to two generators one generating the samples and the other providing the aspect for that sample. In this work, the BEGAN implementation has been utilized to train the generator, and the inverse of the generator is implemented using the Mean Absolute Error loss presented in [41]. Based on the nature of the aspect, the aspect generator could be an MLP or a decoder network mapping the latent space to the aspect space.

In Appendix F³ this approach has been applied to face generators while the aspects are the facial landmarks, and also low-quality iris images while the segmentation map is considered as the aspect. Randomly generated faces and their corresponding generated landmarks are shown in figure 3.16 and randomly generated iris images and their corresponding segmentation maps are shown in figure 3.17.

³Code is available at: https://github.com/shababqcd/GAN_Aspect

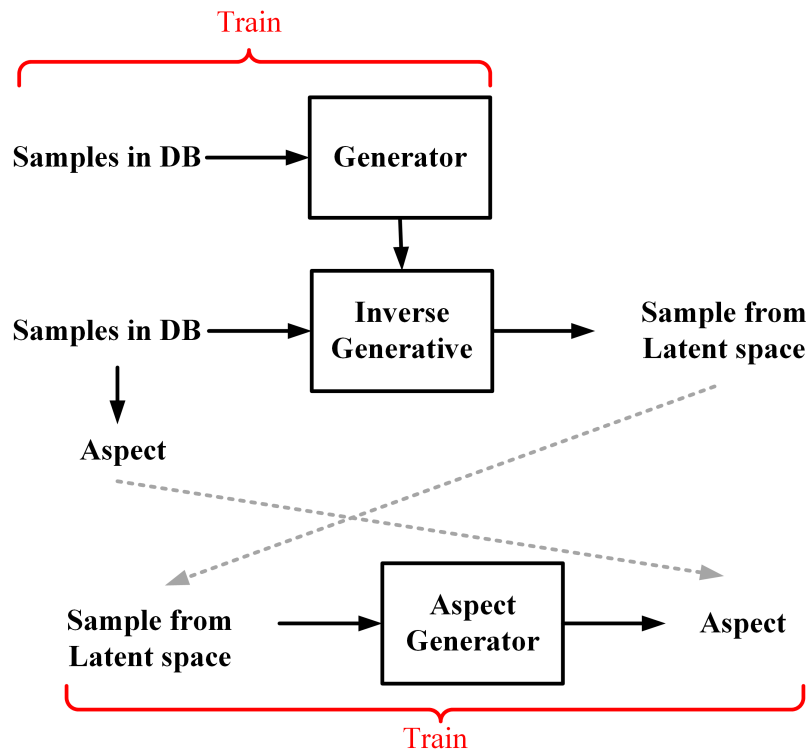


Fig. 3.14 Training procedure presented in Appendix F.

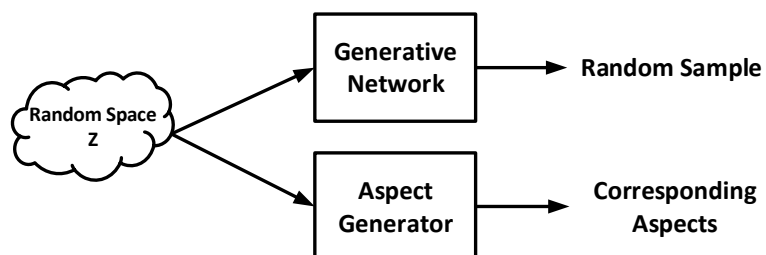


Fig. 3.15 The inference for generating samples alongside their corresponding aspects.

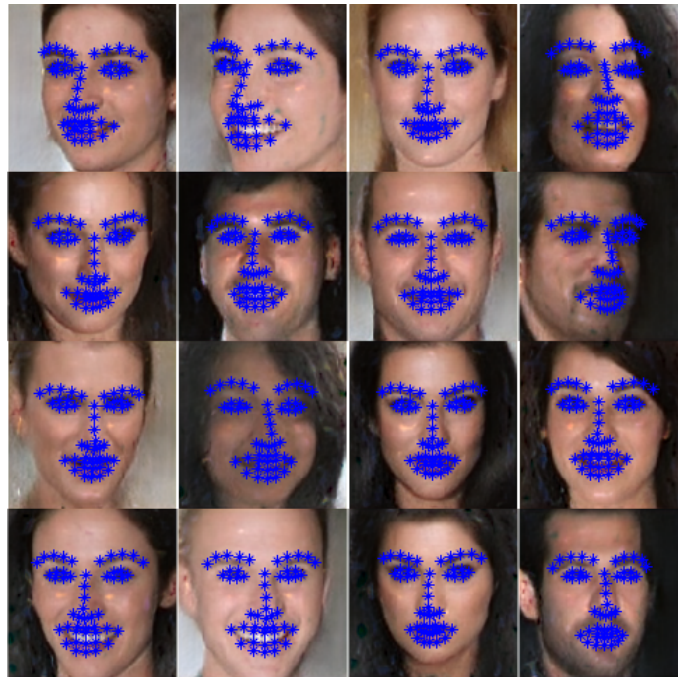


Fig. 3.16 Randomly generated faces and their corresponding landmarks.

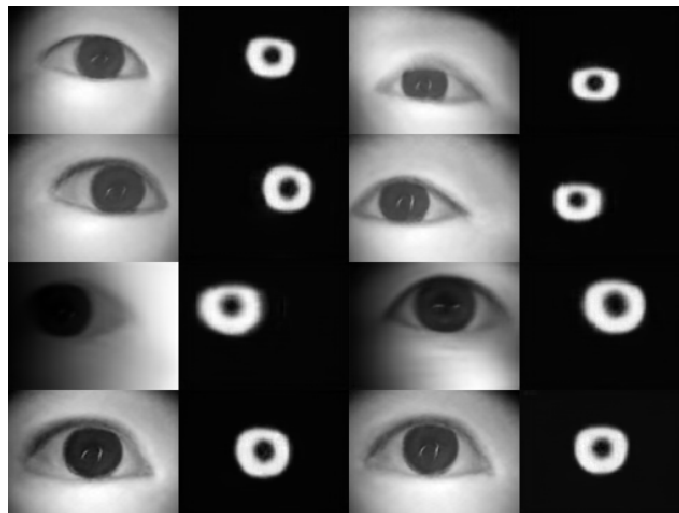


Fig. 3.17 Randomly generated low-quality iris images and their corresponding segmentation maps.

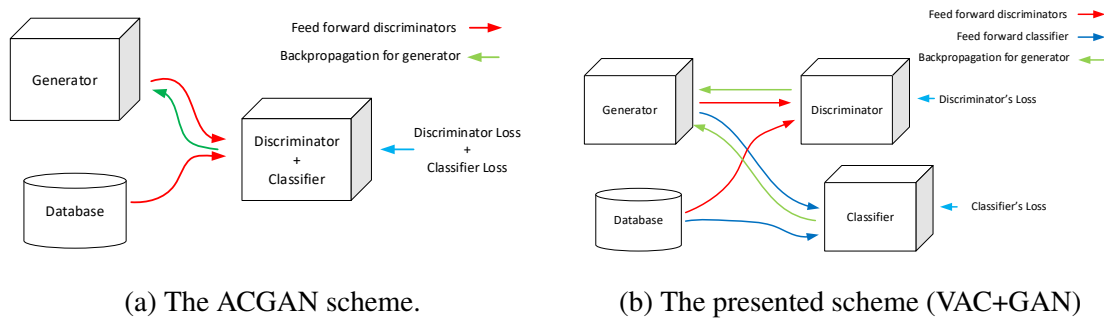


Fig. 3.18 ACGAN vs presented model.

3.3.2 Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN)

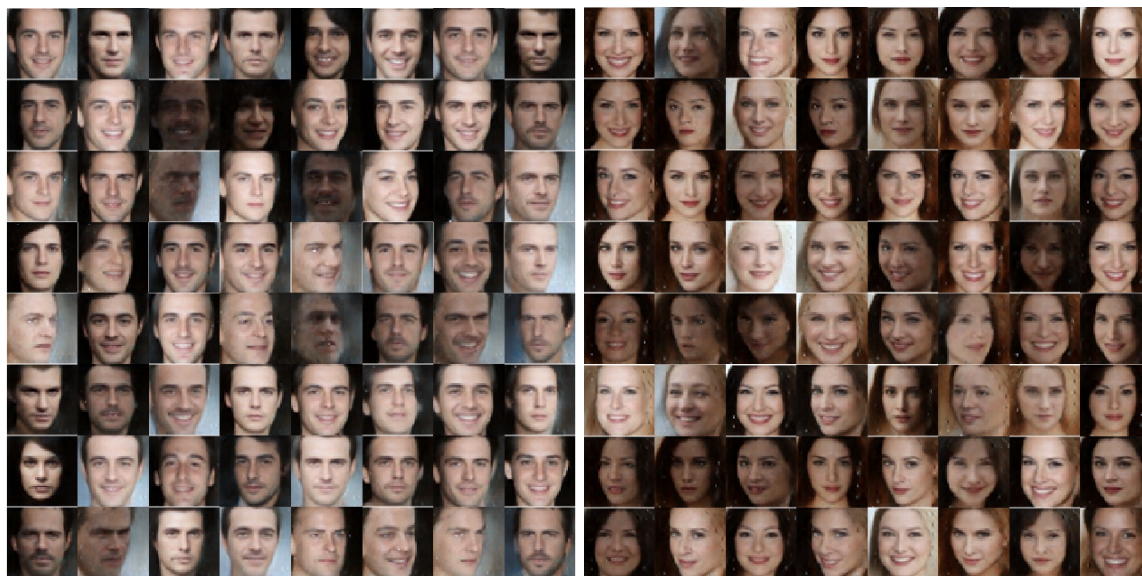
Conditional generators are one of the generator types that produce samples from a specific class given the right input sequences. Training conditional generators are one of the most appealing applications of GAN. With the recent restrictions on data acquisition and manipulation such as GDPR, it is become more important to be able to expand the existing databases and/or generate a new set of anonymous samples from scratch. The conditional generators gives the opportunity to construct data, given a specific class label. These datasets could be used to train classifiers without raising any concern regarding the data acquisition procedure.

Conditional GAN (CGAN) [100] and Auxiliary Classifier GAN (ACGAN) [42] are among the most utilized schemes for this purpose. Wherein the CGAN approach uses the auxiliary class information alongside with partitioning the latent space and ACGAN improves the CGAN idea by introducing a classification loss which back-propagates through the discriminator and generator network. The CGAN method is versatile enough to apply to every variation of GAN. But ACGAN is restricted to a specific loss function which decreases its adaptivity to other GAN varieties.

In the presented scheme [47, 48] (Appendices G⁴ and, H⁵), the ACGAN technique is extended to be applicable to any GAN implementation by placing a classifier in parallel with the discriminator network. The classification loss backpropagates through the generator alongside with the original GAN loss value. Figure 3.18 shows the ACGAN and VAC+GAN techniques in training phase. In these works, it has been shown that using the binary cross-

⁴Code available at: <https://github.com/shababqcd/VAC-GAN/tree/master/VACGANbinary>

⁵Code available at: <https://github.com/shababqcd/VAC-GAN/tree/master/VACGANcifar10> and <https://github.com/shababqcd/VAC-GAN/tree/master/VACGANmnist>



(a) Constrained to generate male samples

(b) Constrained to generate female samples.

Fig. 3.19 Generator trained using the proposed method (VAC+GAN).

entropy (for binary cases) and categorical cross-entropy (for multi-class scenarios) loss functions for classifier increases the Jensen-Shannon divergence between the distribution of different classes in the generator.

The results has been compared to other versatile method known as Conditional GAN (CGAN) for gender specified face generation and also the implementation of CGAN, CD-CGAN and ACGAN on MNIST dataset and also the comparisons are given on CFAR10 dataset with respect to ACGAN method. The ACGAN gives comparable results, but this method cannot be applied to other variations of GAN such as BEGAN.

The main advantage of this technique is its versatility. For example for the face generation application, the VAC+GAN is applied to BEGAN model and for number generation it is employed to DCGAN approach. The results for the gender specified generator are shown in figure 3.19 wherein the generator provides male or female samples given right input sequence and, figure 3.20 shows the results for a multi-class problem which is generating samples for a class in MNIST dataset.

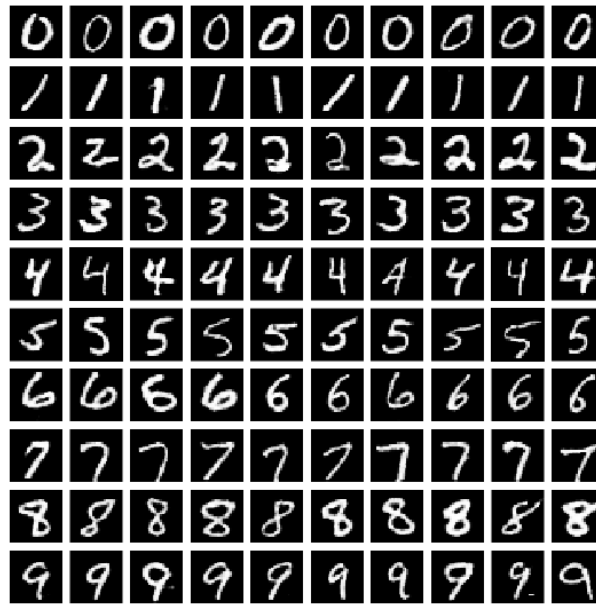


Fig. 3.20 Samples drawn from conditional generator trained using proposed scheme (VAC+GAN) on MNIST dataset. each row corresponds to one class.

3.3.3 Versatile Auxiliary Regressor with Generative Adversarial Network (VAR+GAN)

In the previous section, the proposed method trains a generator able to produce samples from discrete classes. For example, it produces male or female faces given proper input sequence. But the question is if it is possible to train conditional generators for continuous aspects. for example, giving a pose or landmark positions to the generator and it produces faces fixed to those landmarks. In this work, a new framework known as Versatile Auxiliary Regressor with Generative Adversarial Network (VAR+GAN) is presented which is able to train such conditional generators.

The idea of VAR+GAN [49](Appendix I⁶) is to place a regression network in parallel with the discriminator network and backpropagate the regression error through the generator (figure 3.21). In this approach, one can apply restrictions on the generated data to be limited to some continuous constraints. For example, the generator creates faces given a specific set of landmarks using the VAR+GAN idea. The other similar idea is conditional Bi-directional GAN (cBiGAN) [101] which is a mixture of CGAN [100] and Bidirectional GAN (BiGAN) [102]. See figure 3.22. The main issue with this method is the lack of adaptability to every

⁶Code available at: <https://github.com/shababqcd/VAR-GAN>

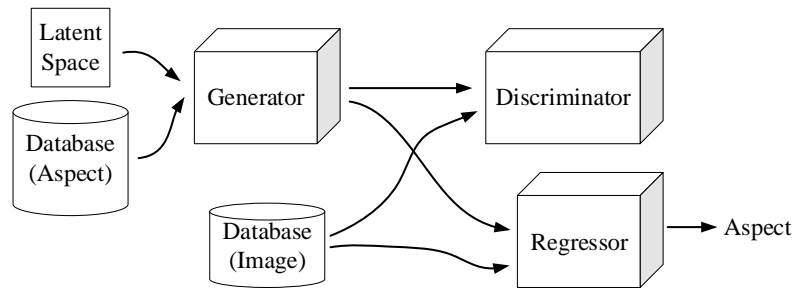


Fig. 3.21 The error from the regression network backpropagates through the generator in VAR+GAN framework.

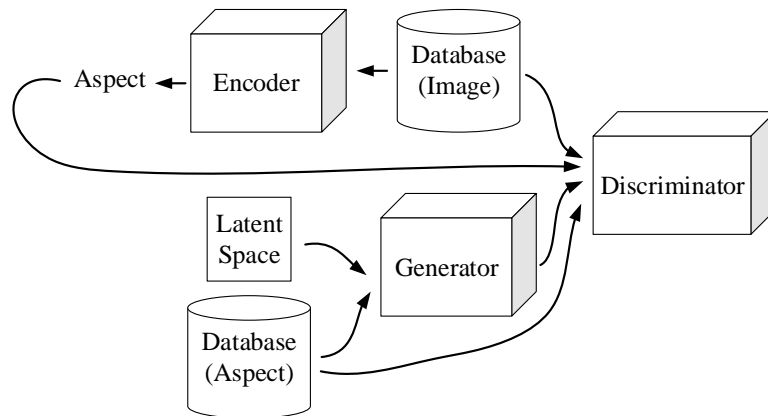


Fig. 3.22 The cBiGAN framework merges CGAN and BiGAN approaches in a single model.

GAN variation. This method is only applicable to BiGAN scheme.

The main advantage of the proposed method is that it applies to any GAN variation. This gives the opportunity of utilizing different GAN implementations since each of them are designed for some specific problem.

The other main contribution of the VAR+GAN technique is its particular loss function designed for the regression network. This loss function is inspired by the categorical cross-entropy loss function and is given by

$$L_R = \int \int dp_z(\mathbf{z}) \left(-\log \left(1 - (y - R(G(\mathbf{z}))) \right) \right) dz \quad (3.3)$$

wherein \mathbf{z} is the latent space variable, $dp_z(\mathbf{z})$ is the distribution of an infinitesimal partition of latent space, y is the target variable (ground truth), R is the regression function and G is the generator function.

This in [49](Appendix I) we show that this loss function guarantees to decrease the entropy of the generator's output while increasing the Jensen Shannon Divergence of two sets of outputs correspond to any two different aspect sets.

The output of the network trained on faces and their landmarks are shown in figure 3.23 for two different sets of landmarks. Again this method gives a powerful tool to generate a set of anonymous samples for any desired aspect set. The other application of this method is to expand a dataset for rare aspects. For example if a dataset lacks some specific poses and landmarks, one can generate many samples for those poses and add to database. This leads to training better regression solutions in future.

3.4 Conclusion

With the emergence of fast and efficient hardware to train and infer Deep Neural Networks, they become the most popular approach to solve machine learning problems in recent years. There has been an immense amount of research on this topic in the last few years and this thesis targets three fields in the Deep Learning science.

1. Network Design: For a given problem finding an optimum network structure is a nontrivial task. There are several factors including the depth of the network, layer type and non-linearity type that should be determined in the design procedure. But

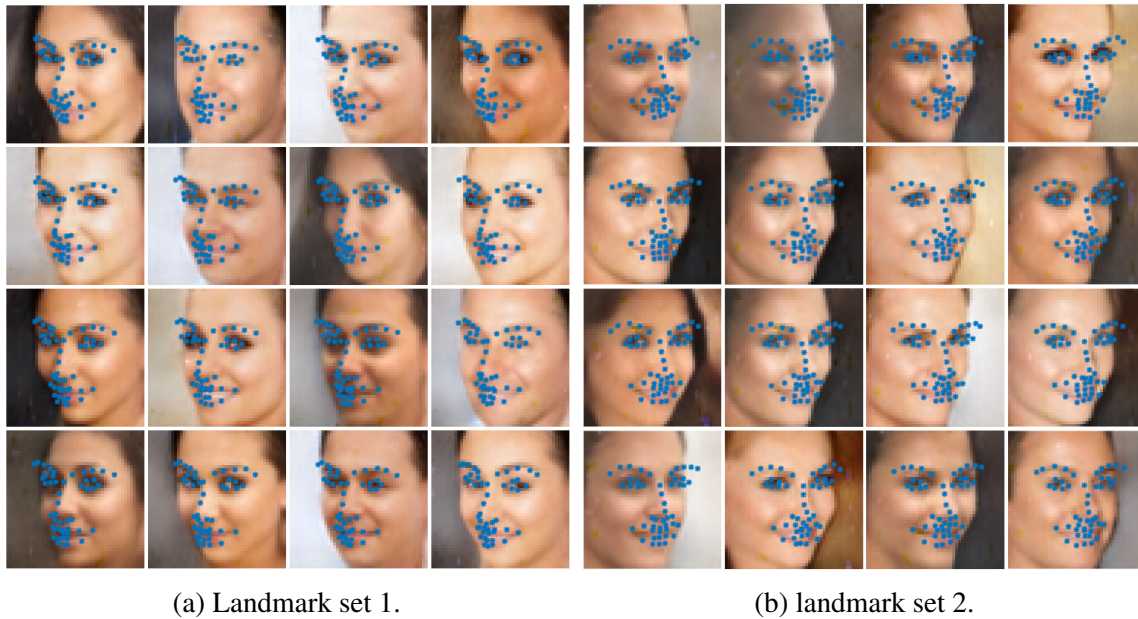


Fig. 3.23 Generator outputs for proposed method (VAR+GAN) given particular landmarks

for any given problem one can find several successful deep learning solutions. In this work, a new method known as Semi Parallel Deep Neural Network (SPDNN) is presented which merges several network architectures into a single model using the graph contraction method. Many observations including iris segmentation and monocular depth estimation problems show the effectiveness of the SPDNN method. This method also reduces the number of the trainable parameters for the final model compared to the mixture of every parent network. For example, for the iris segmentation task [2] (Appendix D), we got 33% reduction of combined network size while for the Monocular Depth Estimation [1] (Appendix B) network this number goes up to 75% reduction in network size. Also in [45] (Appendix A) it has been shown even by keeping the network size same as each parent network, the SPDNN gives better results in the iris segmentation task. SPDNN might not solve all the network design problems entirely but it is a step forward in this field and also helps to take advantage of several designs simultaneously.

2. Data augmentation: The biggest lesson learned in this Ph.D. course was the value of data in Deep Learning. The data define every single aspect of the problem and address how the problem should be solved. Reorienting the data to be suitable for a particular task is known as data augmentation. It is usually done blindly by most of the Deep Learning scientists, such as flipping, rotating and adding noise to the data which in the most of the cases helps to generalize the solution but by knowing the nature of the

problem, the augmentation process could be designed by an expert. For example in this thesis, a DNN is trained to segment the iris region out of the low-quality eye socket image. In order to imitate the low-quality nature of the customer level iris acquisition handheld devices, several data processing steps have been implied including, contrast reduction, shadowing and motion blur. The results of this augmentation approach alongside with the SPDNN design gives the best segmentation accuracy for low-quality databases such as UBIRIS, and MobBio reported in the literature (By the time of writing this thesis).

In some cases, the nature of the problem is vague or unknown. This is happening when the implementation scenarios are not well defined. This uncertainty makes the augmentation process more difficult. One of the best solutions to expand the database in these cases is to merge several samples to produce a new representation. The Smart Augmentation technique presented in this thesis is a method that learns how to mix and blend several samples of one class to produce a new sample of the same class while at the same time the generated sample should reduce the classification loss of a second network. Smart Augmentation has been tested on more than 30 scenarios and it has been shown that this methodology decreases overfitting, increases generalization capability and can significantly reduce the number of parameters required to perform the same task (i.e., a smaller network can work just as well as a large network).

3. Deep Generators: Being able to learn the data distribution and draw randomly generated samples has many appealing applications in data-driven machine learning methods. In this thesis, a method is presented to generate samples alongside any aspect of the sample. For example, a random face is generated and the facial landmarks for that face are also generated. This is done by training two generators; one for the actual samples and the other for the aspect.

Another method is presented known as Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN). In this framework, a conditional generator is trained which is able to produce samples drawn from a desired class. For example, it draws randomly generated male faces for some specific input sequence and female samples for a different sequence. This expands a dataset by adding anonymous samples suitable for classification tasks. The advantage of the presented method to other conditional generators is its versatility. It basically is applicable to every variant of GAN implementations.

The other contribution is an expansion of (VAC+GAN) method known as Versatile Auxiliary Regressor with Generative Adversarial Network (VAR+GAN). In this method,

the generator is constrained to a continuous aspect of the data. For example, this generator is able to produce random facial images that are fixed to a specific pose or a set of landmarks. This method is versatile as well. It means that it is simply applicable to any variant of GAN implementations. This method helps to expand the size of the database by producing anonymous samples following a specific continuous aspect. The other application is to produce samples with rare aspects. Like generating faces in some specific pose or intermediate poses between two samples. This certainly helps to solve the regression problems in more general scenarios.

3.5 Future Works

The future works include mixing the VAR+GAN and VAC+GAN ideas to train conditional generators which accept discrete and continuous aspects at the same time. For example, in the face generation case one can generate random faces which belong to a specific gender class and also are fixed onto a particular landmark set.

The other idea is to find a way to inject more information during image generation in order to train a generator for a specific task. In other words in all the GAN ideas so far, the generator is producing extra samples without looking at a specific problem. One of the ideas for future work is training generators that produce samples for a specific problem. This problem can be solved using photo-realistic style transfer as well.

For the Smart Augmentation project, the future works include multi class implementation of the idea and also mixing it with GAN idea to generate more realistic images.

The future work for the iris segmentation project is to train a network to segment low quality, consumer level, off-axis iris images which are useful in AR/VR applications.

References

- [1] S. Bazrafkan, H. Javidnia, J. Lemley, and P. Corcoran, “Semiparallel deep neural network hybrid architecture: first application on depth from monocular camera,” *Journal of Electronic Imaging*, vol. 27, pp. 27 – 27 – 19, 2018. [Online]. Available: <https://doi.org/10.1117/1.JEI.27.4.043041>
- [2] S. Bazrafkan, S. Thavalengal, and P. Corcoran, “An end to end deep neural network for iris segmentation in unconstrained scenarios,” *Neural Networks*, vol. 106, pp. 79 – 95, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S089360801830193X>
- [3] J. Lemley, S. Bazrafkan, and P. Corcoran, “Smart augmentation learning an optimal data augmentation strategy,” *IEEE Access*, vol. 5, pp. 5858–5869, 2017.
- [4] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural networks*, vol. 61, pp. 85–117, 2015.
- [5] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.
- [6] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.
- [7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.
- [8] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
- [9] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3730–3738.
- [10] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *Advances in neural information processing systems*, 2015, pp. 649–657.
- [11] J. Bergstra, F. Bastien, O. Breuleux, P. Lamblin, R. Pascanu, O. Delalleau, G. Desjardins, D. Warde-Farley, I. Goodfellow, A. Bergeron *et al.*, “Theano: Deep learning on gpus with python,” in *NIPS 2011, BigLearning Workshop, Granada, Spain*, vol. 3. Citeseer, 2011.

- [12] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. Sønderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. D. Fauw, M. Heilman, D. M. de Almeida, B. McFee, H. Weideman, G. Takács, P. de Rivaz, J. Crall, G. Sanders, K. Rasul, C. Liu, G. French, and J. Degraeve, “Lasagne: First release.” Aug. 2015. [Online]. Available: <http://dx.doi.org/10.5281/zenodo.27878>
- [13] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: A system for large-scale machine learning.” in *OSDI*, vol. 16, 2016, pp. 265–283.
- [14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.
- [15] A. Paszke, S. Chintala, R. Collobert, K. Kavukcuoglu, C. Farabet, S. Bengio, I. Melvin, J. Weston, and J. Mariethoz, “Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration, may 2017.”
- [16] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, “Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems,” *arXiv preprint arXiv:1512.01274*, 2015.
- [17] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.
- [18] S. Lawrence, C. L. Giles, and A. C. Tsoi, “What size neural network gives optimal generalization? convergence properties of backpropagation,” Tech. Rep., 1998.
- [19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [20] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European conference on computer vision*. Springer, 2014, pp. 818–833.
- [21] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.
- [22] F. Luan, S. Paris, E. Shechtman, and K. Bala, “Deep photo style transfer,” *CoRR*, *abs/1703.07511*, vol. 2, 2017.
- [23] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [24] A. Deshpande, “The 9 deep learning papers you need to know about (understanding cnns part 3),” 2016. [Online]. Available: <https://adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html>
- [25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich *et al.*, “Going deeper with convolutions.” *Cvpr*, 2015.

- [26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [27] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [28] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [29] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.
- [30] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016, pp. 379–387.
- [31] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.
- [33] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," *arXiv preprint*, 2017.
- [34] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.
- [35] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 142–158, 2016.
- [36] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv preprint arXiv:1511.00561*, 2015.
- [37] V. Badrinarayanan, A. Handa, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," *arXiv preprint arXiv:1505.07293*, 2015.
- [38] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [39] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017, pp. 214–223.
- [40] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

- [41] D. Berthelot, T. Schumm, and L. Metz, “Began: Boundary equilibrium generative adversarial networks,” *arXiv preprint arXiv:1703.10717*, 2017.
- [42] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier gans,” *arXiv preprint arXiv:1610.09585*, 2016.
- [43] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
- [44] hindupuravinash, “The gan zoo,” 2018. [Online]. Available: <https://github.com/hindupuravinash/the-gan-zoo>
- [45] S. Bazrafkan and P. M. Corcoran, “Pushing the ai envelope: merging deep networks to accelerate edge artificial intelligence in consumer electronics devices and systems,” *IEEE Consumer Electronics Magazine*, vol. 7, no. 2, pp. 55–61, 2018.
- [46] S. Bazrafkan, T. Nedelcu, P. Filipczuk, and P. Corcoran, “Deep learning for facial expression recognition: A step closer to a smartphone that knows your moods,” in *Consumer Electronics (ICCE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 217–220.
- [47] S. Bazrafkan, H. Javidnia, and P. Corcoran, “Versatile auxiliary classifier with generative adversarial network (vac+gan),” *arXiv preprint arXiv:1805.00316*, 2018.
- [48] S. Bazrafkan and P. Corcoran, “Versatile auxiliary classifier with generative adversarial network (vac+ gan), multi class scenarios,” *arXiv preprint arXiv:1806.07751*, 2018.
- [49] —, “Versatile auxiliary regressor with generative adversarial network (var+gan),” *arXiv preprint arXiv:1805.10864*, 2018.
- [50] J. Schmidhuber, “Who invented backpropagation?” 2014.
- [51] C. Davis. (2017, June) Why is deep learning so easy? [Online]. Available: <https://www.quora.com/Why-is-deep-learning-so-easy>
- [52] N. Tishby, F. C. Pereira, and W. Bialek, “The information bottleneck method,” *arXiv preprint physics/0004057*, 2000.
- [53] N. Tishby and N. Zaslavsky, “Deep learning and the information bottleneck principle,” in *Information Theory Workshop (ITW), 2015 IEEE*. IEEE, 2015, pp. 1–5.
- [54] R. Shwartz-Ziv and N. Tishby, “Opening the black box of deep neural networks via information,” *arXiv preprint arXiv:1703.00810*, 2017.
- [55] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, “On the information bottleneck theory of deep learning,” in *International Conference on Learning Representations*, 2018.
- [56] Aaron. (2018, February) Failure to replicate schwartz-ziv and tishby. [Online]. Available: https://planspace.org/20180213-failure_to_replicate_schwartz-ziv_and_tishby/

- [57] P. Lewis, "Analogy between human and artificial neural nets," *Acesso em Setembro de*, 2014.
- [58] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.
- [59] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.
- [60] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.
- [61] Y. LeCun *et al.* (2015) Lenet-5, convolutional neural networks. [Online]. Available: <http://yann.lecun.com/exdb/lenet>
- [62] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [63] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, p. 947, 2000.
- [64] A. Sharma. (2017, March) Understanding activation functions in neural networks. [Online]. Available: <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>
- [65] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.
- [66] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
- [67] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.
- [68] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.
- [69] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 972–981.
- [70] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan, "Deep learning with s-shaped rectified linear activation units." in *AAAI*, 2016, pp. 1737–1743.
- [71] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: a self-gated activation function," *arXiv preprint arXiv:1710.05941*, 2017.
- [72] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
- [73] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2018.

- [74] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning.” in *AAAI*, vol. 4, 2017, p. 12.
- [75] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [76] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.
- [77] L. Rosasco, E. D. Vito, A. Caponnetto, M. Piana, and A. Verri, “Are loss functions all the same?” *Neural Computation*, vol. 16, no. 5, pp. 1063–1076, 2004.
- [78] C. Godard, O. Mac Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in *CVPR*, vol. 2, no. 6, 2017, p. 7.
- [79] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [80] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.
- [81] F. E. Curtis and K. Scheinberg, “Optimization methods for supervised machine learning: From linear models to deep learning,” in *Leading Developments from INFORMS Communities*. INFORMS, 2017, pp. 89–114.
- [82] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.
- [83] A. S. Walia. (2017, June) Types of optimization algorithms used in neural networks and ways to optimize gradient descent. [Online]. Available: <https://goo.gl/q3wosS>
- [84] V. Bushaev. (2017, December) Stochastic gradient descent with momentum. [Online]. Available: <https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d>
- [85] Y. Nesterov, “A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$,” in *Doklady AN USSR*, vol. 269, 1983, pp. 543–547.
- [86] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.
- [87] M. D. Zeiler, “Adadelta: an adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012.
- [88] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSEERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.
- [89] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

- [90] A.-S. Ungureanu, S. Bazrafkan, and P. Corcoran, “Deep learning for hand segmentation in complex backgrounds,” in *Consumer Electronics (ICCE), 2018 IEEE International Conference on*. IEEE, 2018, pp. 1–2.
- [91] P. Y. Simard, D. Steinkraus, J. C. Platt *et al.*, “Best practices for convolutional neural networks applied to visual document analysis.” in *ICDAR*, vol. 3, 2003, pp. 958–962.
- [92] F. Chollet *et al.*, “Keras,” <https://github.com/fchollet/keras>, 2015.
- [93] S. Thavalengal, P. Bigioi, and P. Corcoran, “Iris authentication in handheld devices—considerations for constraint-free acquisition,” *IEEE Transactions on Consumer Electronics*, vol. 61, no. 2, pp. 245–253, 2015.
- [94] ———, “Evaluation of combined visible/nir camera for iris authentication on smartphones,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 42–49.
- [95] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, “Improving the robustness of deep neural networks via stability training,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4480–4488.
- [96] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.
- [97] H. Valpola and J. Karhunen, “An unsupervised ensemble learning method for nonlinear dynamic state-space models,” *Neural computation*, vol. 14, no. 11, pp. 2647–2692, 2002.
- [98] K. Frans, “Variational autoencoders explained,” 2016. [Online]. Available: <http://kvfrans.com/variational-autoencoders-explained/>
- [99] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” *arXiv preprint arXiv:1601.06759*, 2016.
- [100] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014.
- [101] K. Wang, R. Zhao, and Q. Ji, “A hierarchical generative model for eye image synthesis and eye gaze estimation.”
- [102] A. Jaiswal, W. AbdAlmageed, Y. Wu, and P. Natarajan, “Bidirectional conditional generative adversarial networks,” *arXiv preprint arXiv:1711.07461*, 2017.

Appendix A

Pushing the AI Envelope: Merging deep networks to accelerate edge artificial intelligence in consumer electronics devices and systems

Pushing the AI Envelope



©ISTOCKPHOTO.COM/KTSIMAGE

Merging deep networks to accelerate edge artificial intelligence in consumer electronics devices and systems.

By Shabab Bazrafkan and Peter Corcoran


DEEP NEURAL NETWORKS (DNNs) ARE WIDELY USED by both academic and industry researchers to solve many long-standing problems in machine learning. There has been such a growth of research in this field, and it has been applied to so many varying problems, that it would be accurate to say that we may be living through the precursor of the singularity [1]. But regardless of one's views on artificial intelligence (AI), there is no doubt that there is a wealth of recent research that leverages the use of various DNNs to solve a broad range of pattern recognition and classification problems. Examples range from the introduction of smart speakers with intelligent assis-

nants to the application of DNNs to solve recalcitrant problems in computer vision for autonomous vehicles. Many of these problems can have very useful applications in the design of smarter consumer electronics (CE) systems and devices. The question for CE engineers is how to leverage this wealth of academic and industry research efforts, turning them into practical DNN solutions suitable for deployment in practical devices and electronic systems.

FINDING THE FOREST

There are a number of challenges here. First, academic research is typically focused on a specific facet of a research problem so that the authors can compare their results with different approaches to the same problem, thereby trying to show significant improvement in certain aspects of performance, accuracy, or robustness.

Digital Object Identifier 10.1109/MCE.2017.2775245
Date of publication: 8 February 2018



The engineer, approaching this problem from a broader perspective, would like to be able to combine multiple DNN techniques, leveraging the positive benefits of each.

Rarely does a new technique or tactic perform so well that all previous methods become obsolete. However, the engineer, approaching this problem from a broader perspective, would like to be able to combine multiple DNN techniques, leveraging the positive benefits of each and mitigating their weaker aspects.

A second challenge is that academic research is rarely concerned with optimizing a technique for size or architectural efficiency. Typically, many resources are directed to the narrow problem at hand, and even where optimizations are investigated—and, thankfully, the deep-learning research community has become more sensitive to their importance—they apply to only a particular DNN architecture. In general, researchers are focused on designing a network to solve an immediate problem and do not think more widely about sharing structures or resources between multiple DNNs.

Thus, for the engineer approaching a particular problem in machine learning and seeking to adopt techniques from the literature, there will typically be a handful of different DNN solutions available, each adopting different network structures and optimizations while still sharing many common network layers and data inputs. However, there has been little research concerned with techniques for merging and optimizing a combination of existing DNN approaches into a more holistic solution. In effect, almost everyone seems to be focused on growing an insular cluster of trees and developing these as quickly as possible, rather than working with others to develop and manage a forest where everyone can take advantage of economies of scale.

Fortunately, most DNN architectures do share many common elements, just as many parts of a forest share similar trees. Furthermore, it is possible to bring such elements together and develop more accurate, optimized, and improved DNNs that will deliver better performance and economies of scale with efficient implementations that are well suited for CE deployments. This is what we will focus on in the remainder of this article.

A CRASH COURSE IN DEEP NETWORKS

Here, we provide a quick refresher on some of the most important elements of a DNN. For those who are unfamiliar with deep learning, our previous article [2] is recommended. DNN networks comprise several signal-processing methods applied in sequence to an input data structure. In image-understanding tasks, these signal-processing elements are typically convolutional and fully connected layers, followed by pooling

and regularization layers. Data flow through these layers in sequence, eventually emerging in a modified structure that typically enables a binary interpretation of some feature or characteristic of the input data structure. Typically, the chain of data processing layers is much longer (deeper) than in classical neural networks, hence the descriptive term *deep* that is applied to such networks.

THE MAIN CATEGORIES OF DEEP NETWORK LAYERS

Deep networks comprise a succession of data-processing layers, each taking inputs from a preceding layer and filtering a set of inputs to generate a set of outputs.

▼ *Convolutional layers*: These are the best-known layers; they convolve the input data from the previous layer I with a kernel W . In the general case, an offset bias b is added to this convolution as follows:

$$P = I * W + b.$$

In computer-vision applications, the original input I is typically a structured set of image pixels, but as the original data element is processed by successive layers of the network, it becomes altered and transformed until, eventually, a much simpler output is obtained from the network output. Often, this is a simple binary decision, although other forms of output can also be generated.

Other layer types are also used in deep network architectures, such as the following.

▼ *Pooling layers*: These typically apply a nonlinear transform on the input data that is normally used to reduce the size of the input data. Such layers can be thought of as data concentrator elements, and they are important for controlling the overall size of a network, which could grow too large if only convolutional layers were used.

▼ *Fully connected layers*: These are exactly the same as classical neural network layers, where all of the neurons in a layer are connected to all of the neurons in the subsequent layer. The neurons give the summation of their input multiplied by their weights, which is then passed through their activation functions. Even more than with convolutional layers, these can cause the network size to grow, and so, typically, only one or two fully connected layers will be used in most deep networks.

▼ *Regularization layers*: These are used to prevent overfitting inside the network. Different kinds of regularizations have been proposed, the most important ones being 1) weight regularization, 2) dropout regularization, where some data are skipped, and 3) batch normalization, where output data are averaged across several input data. Each of these regularization techniques has advantages and drawbacks that make them more (or less) suitable for specific applications.

THE COSTS OF GOING DEEP

As was previously mentioned, there has been spectacular growth in research on AI in general and on the application

of deep networking techniques in particular. So much so that, faced with almost any contemporary machine-learning problem, it is almost inevitable that one can find in the literature a plethora of varying network architectures derived from a number of core data sets. Typically, each data set is developed and annotated for this specific class of problem, and each DNN derived from these data sets has some particular advantages and drawbacks. The challenge for engineers is to find the best possible network for a specific problem or design goal. But, in practical use cases, there is often no single champion network, and it may be desirable to use several different networks that can provide complementary outputs.

However, much of the recent literature improves on performance aspects by adding layers to deepen the network. While this may improve specific aspects of network performance, a price is paid in terms of additional memory requirements to store the network and extra compute cycles to process the added layers, or a greater area of silicon if the goal is to provide a chipset implementation. Additionally, if the design engineer wishes to combine several of these deep networks in parallel, the resource requirements quickly lead to impractical, even infeasible, solutions.

Naturally, it is possible to go back to the drawing board and design a completely new network from scratch, but designing, implementing, testing, and optimizing deep networks is challenging and time consuming. The same can be said for the creation of new annotated data sets to enhance and focus network capabilities. It would be a great benefit if it were possible to leverage the work of other researchers to obtain improved networks without having to go back to the drawing board for each new problem that comes across the engineer's desk.

FROM MANY, ONE (NETWORK TO RULE THEM ALL)

It was this line of thinking that led to the work we next describe. Note that this article only provides a top-level overview of the technique, but, fortunately, one can find more detailed guidelines on the applications of these methods for a number of different contemporary image analysis problems [3].

Now, suppose there is a set of neural networks designed for a specific task. As an example, suppose that, for a specific deep-learning problem, one can find in the literature N different successful networks, each of which provides reasonable results on that specific task. Each, however, also has its own drawbacks and would fail on some input data. As previously discussed, it would not make sense to implement multiple parallel networks because of the large resource requirements. It is known that deep networks often share some common layers near the output or

There has been little research concerned with techniques for merging and optimizing a combination of existing DNN approaches into a more holistic solution.

input ends of the network, but it will still be necessary to duplicate the bulk of each network to run them in parallel. In addition, there will be a need for a final fully connected network layer, or a convolutional layer to map all of these parallel networks into a concluding output from the DNN (see Figures 1 and 2). It would be convenient to have a methodology to extract a single deep-network architecture from a selection of parallel networks and to be able to further refine and optimize this newly derived architecture across the original training data sets. This is what we are going to outline next.

LET'S TRY AN EXAMPLE

To understand our methodology, it is best to provide a practical working example. In fact, the original iteration of this methodology was somewhat accidental and the results unexpected. When we first applied it, our expectation was that the new architecture derived from several individual DNNs would simply behave as a vote taker.

THE IRIS-SEGMENTATION PROBLEM

As a demonstration, we will consider the iris-segmentation problem on mobile devices. It is well known that iris biometrics has become feasible on mobile devices [4]–[7], and a number of companies have recently added this feature to smartphones [8]. A key issue for the correct operation of the biometric authentication chain on mobile devices is that of

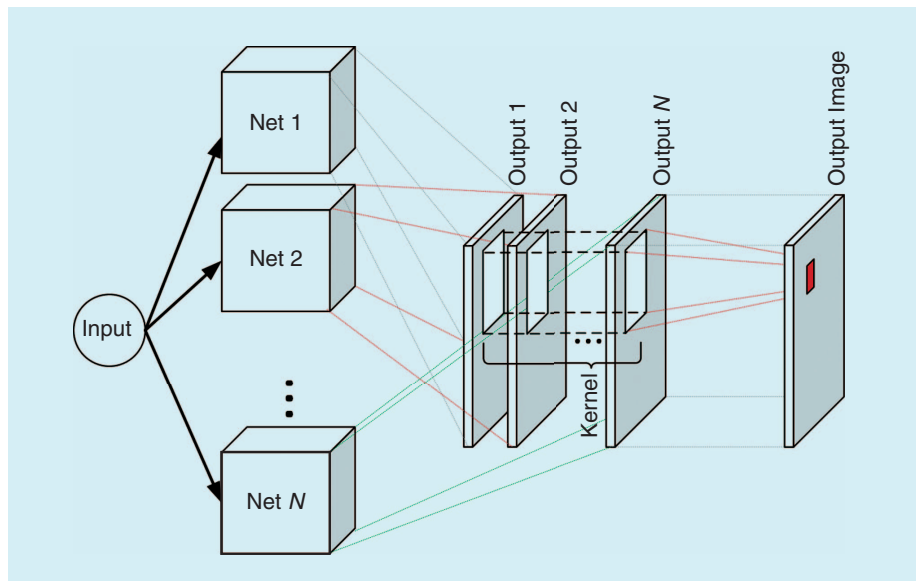


FIGURE 1. The concatenation of the last layer into a single layer (a convolution case).

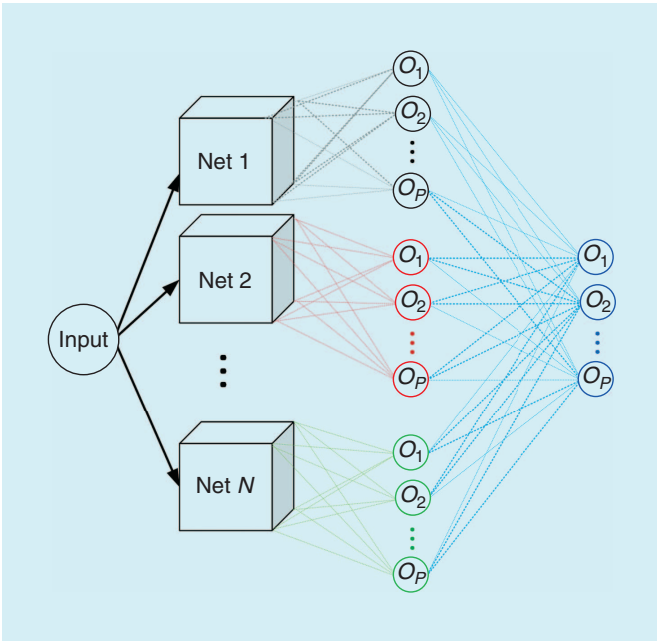


FIGURE 2. The concatenation of the last layer into a single layer (a fully connected case).

iris segmentation [9]–[11]. To address the challenges involved, we derived a large database of poor-quality iris images from the extended CASIA 1000 data set [12]. We augmented the database by reducing the resolution, reducing the contrast inside and outside the iris, and adding motion blur and shadow to each sample [13]. The augmentation code is available online [14]. Then we trained three existing iris-segmentation models on the modified data set.

- 1) The first network was an eight-layer, fully convolutional DNN designed for the iris-segmentation task. All layers used 7×7 kernels. We performed the batch normalization technique after each convolutional layer to avoid overfitting and for faster convergence [Figure 3(a)].
- 2) The second network designed for the problem at hand was a six-layer, fully convolutional network, shown in Figure 3(b). We used a kernel size of 3×3 in all layers. We did not use any pooling in the network, and we employed the batch normalization technique again after each convolutional layer.
- 3) The third proposed network was a five-layer, fully convolutional neural network, shown in Figure 3(c). Here, the kernel

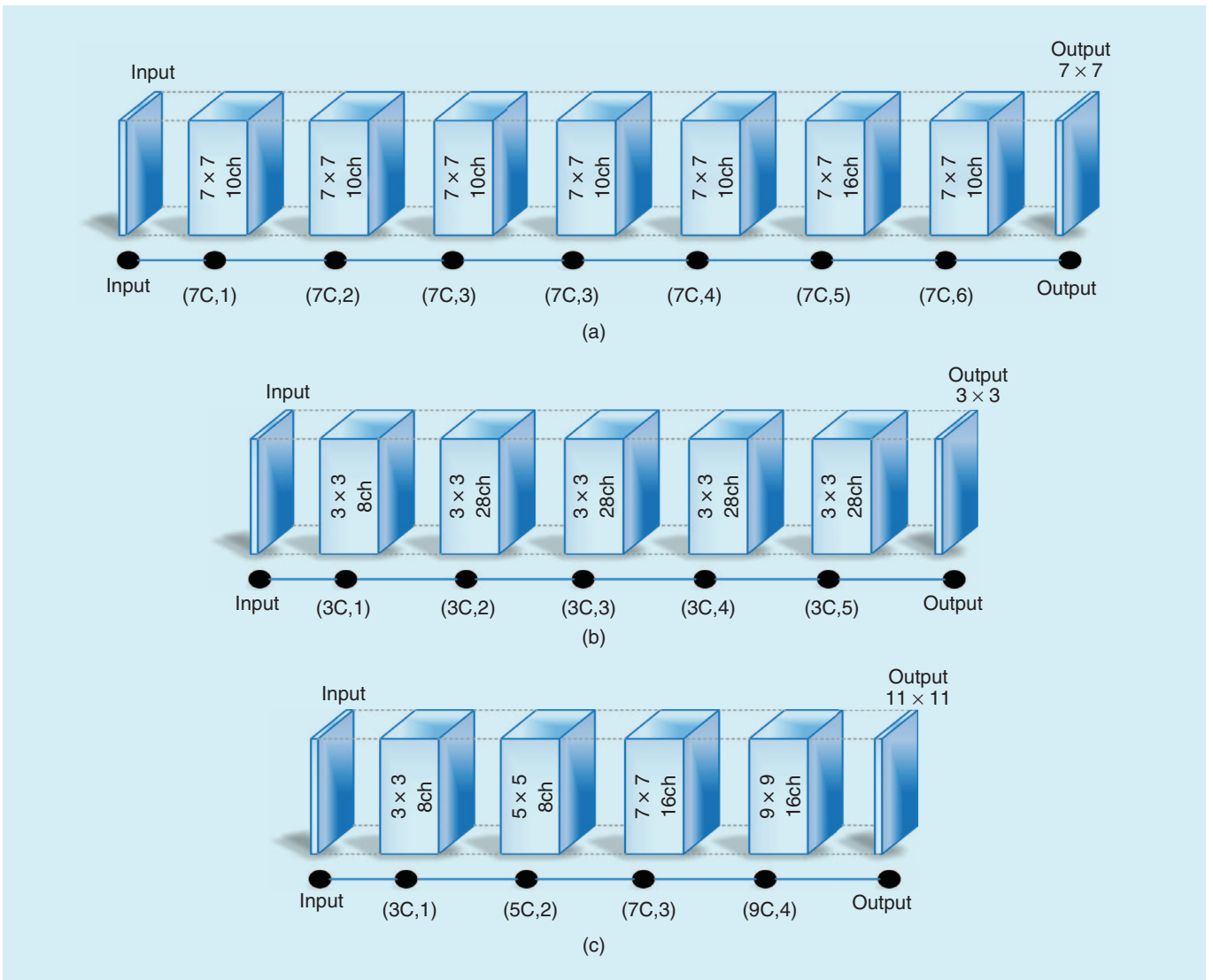


FIGURE 3. The networks we used to perform iris segmentation: (a) network 1, (b) network 2, and (c) network 3. ch: channel.

size increased for each layer, starting with 3×3 for the first layer and 11×11 for the output layer. We did not use any pooling in the network, and again we performed the batch normalization procedure after each convolutional layer.

We trained these three networks using the Nesterov momentum method for the binary cross-entropy loss function on the expanded CASIA 1000 data set.

MERGING THE NETWORKS

The main purpose of the semiparallel DNN (SPDNN) model is to mix and merge several deep architectures and produce a final model that takes advantage of the specialized layers of each architecture but is significantly smaller than the combined sizes of these networks. The approach we adopted is based on graph optimization, and we proceeded as follows.

- 1) We translate each network into its corresponding graph. The graph representation for each network is shown at the bottom in Figure 3(a)–(c).
- 2) We assigned properties to each node.
 - The first property is the operation of the layer and the kernel size: **C** for convolutional, **F** for fully connected layers, and **P** for pooling operation. (Note that, in this simplified example, there are no fully connected and pooling layers; please refer to [3] for more complex examples.) For example, 5C corresponds to a convolutional layer with the 5×5 kernel.

- The second property is the distance of this layer from the input layer. For example, (9C,4) is the label of the node with a 9×9 convolutional operation and having distance 4 from the input data node.
- 3) We placed these graphs in parallel, sharing the same input and output (see Figure 4). We assigned the same label to all nodes with identical properties. In this simple demonstration, all of the nodes with, e.g., the property (7C,3) were assigned label C.
 - 4) We applied the graph contraction operation to this graph. In the graph contraction step, we merged all of the nodes

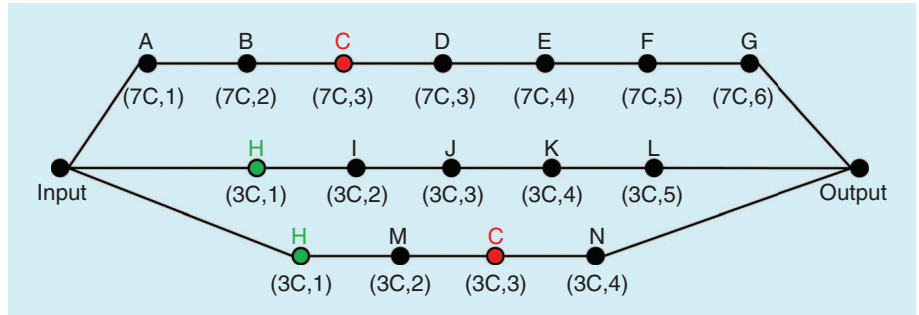


FIGURE 4. All of the graphs in Figure 3 are placed in parallel, sharing a single input and output.

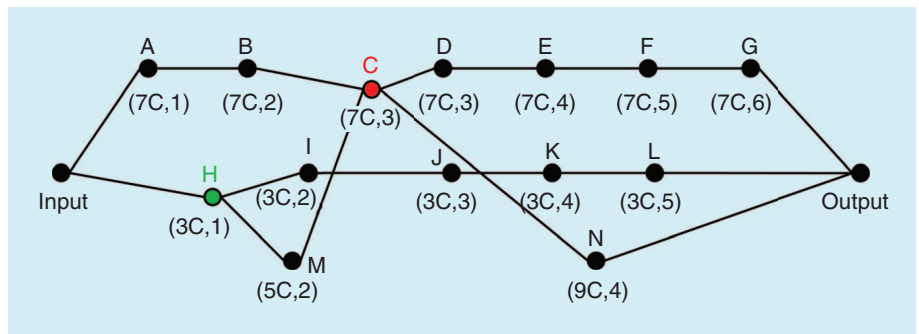


FIGURE 5. The graph contraction is applied to the graph in Figure 4.

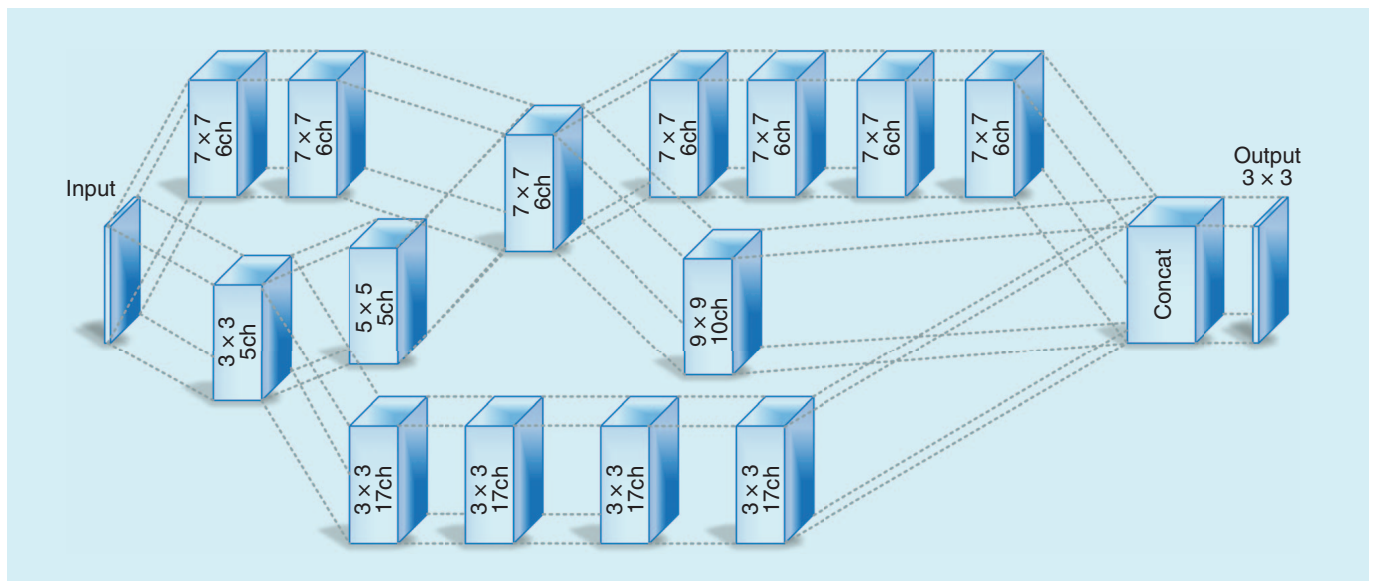


FIGURE 6. The contracted graph is translated into the neural network. Concat: concatenation.

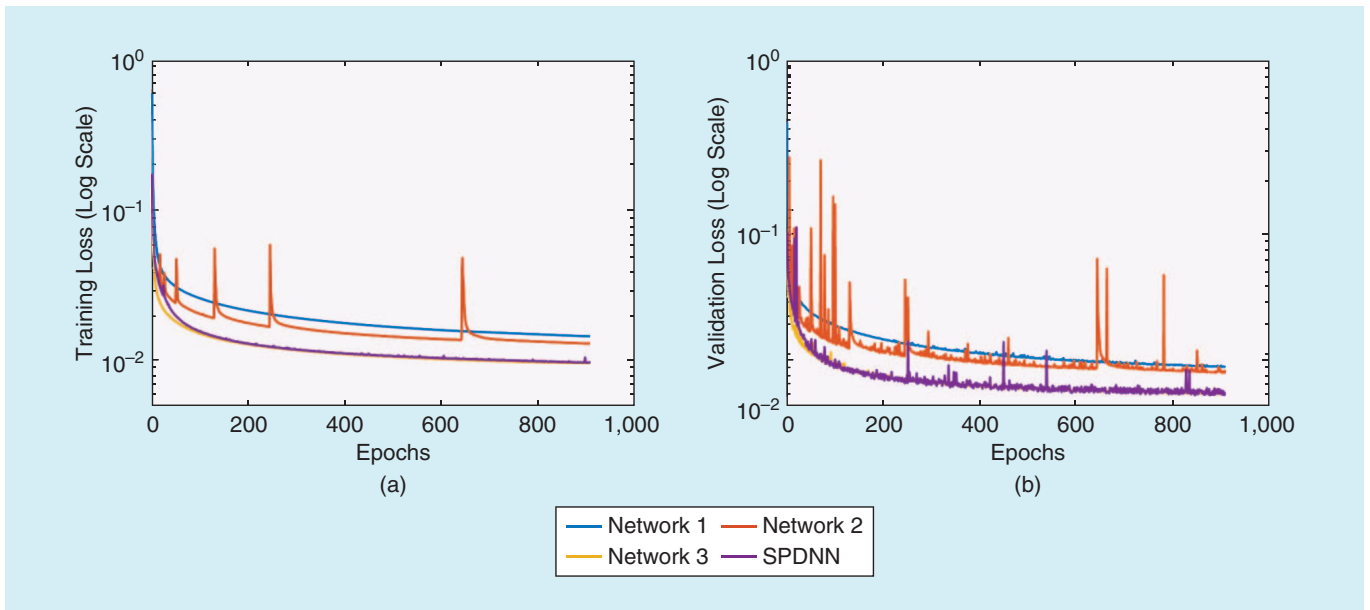


FIGURE 7. (a) The training loss for all of the networks, including the merged (SPDNN) network. (b) The validation loss for all of the networks, including the merged (SPDNN) network.

with the same label into a single node, while saving their connections to the previous and next nodes (see Figure 5).
 5) We translated the graph back into the neural network. Where two or more nodes were merging, the concatenation operation was applied. Additionally, the operation for each node was specified by the first property of the node. The final network is shown in Figure 6.

For the comparisons to be fair, we selected the number of the channels in the final design in a way that the network had

almost the same number of parameters as each of its parent networks. This merged network model was now retrained (using the Nesterov momentum method) and for the same loss function (binary cross-entropy). In this procedure, we employed the data sets used to train the original parent networks. Note that, in this simplified example, we worked with a single data set derived from CASIA, but, in more complex problems, the merged network responded well to cross training with heterogeneous data sets.

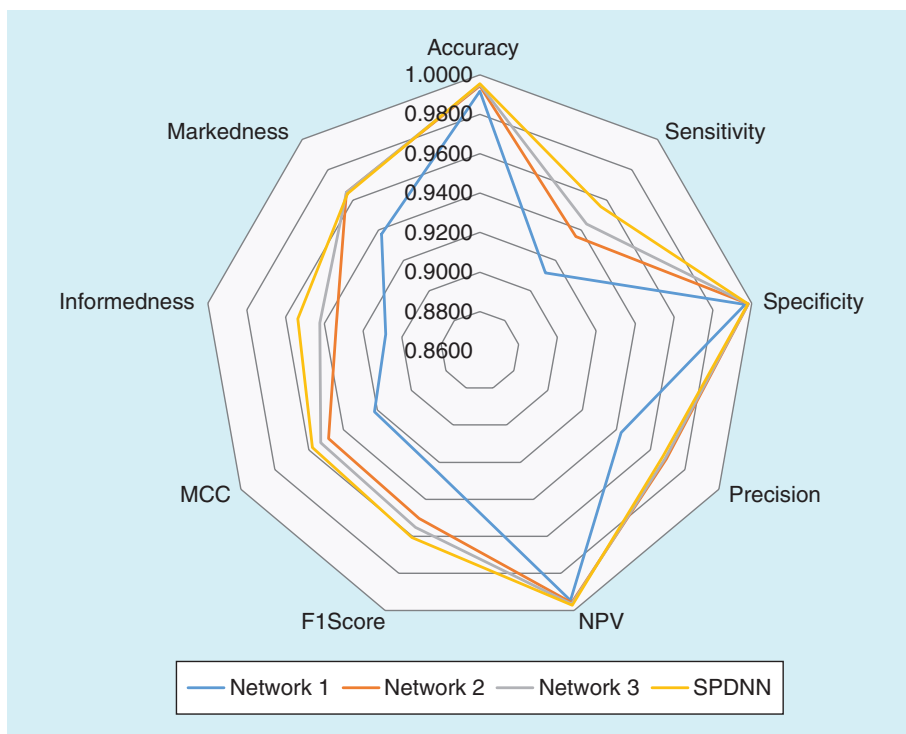


FIGURE 8. The comparisons for all of the networks. (Measurements closer to one indicate better performance.)

As you can see in Figure 7, the losses converged to the best loss of the three networks. This shows that using the SPDNN with several networks helps the model to converge to the best available network. The comparisons on the test data set for all of the networks are shown in Figures 8 and 9. We conclude from these that, although the SPDNN network has the same number of parameters as the original networks 1, 2, and 3, it returns better results in terms of most metrics, including accuracy, sensitivity, negative predictive value (NPV), F1Score, Matthews correlation coefficient (MCC), informedness, and false negative rate (FNR). In particular, the higher accuracy, F1Score, and MCC show that the SPDNN idea offers better performance, in general, compared to its parent networks. Not all metrics are improved over the best of the three networks, but this is a relatively simple application of the methodology.

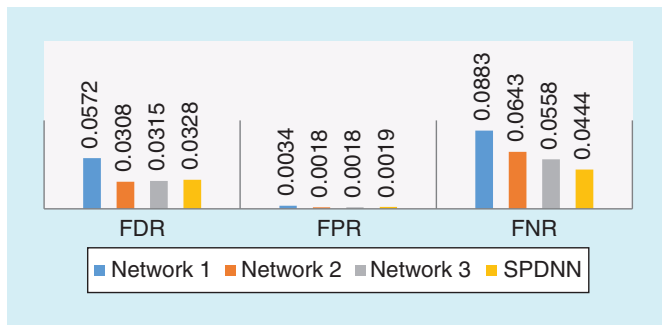


FIGURE 9. The comparisons for all of the networks. (Measurements closer to 0 indicate better performance.) FDR: false discovery rate; FPR: false positive rate.

CONCLUSION

From the training and testing results, we can see that our methodology helped the merged network to achieve an optimal design and increased the convergence in the training stage and in the test phase. In more extensive studies on heterogeneous data sets, we have shown that this methodology can 1) generalize the merged network beyond a trivial combination of parent networks and 2) achieve a significant reduction in size and computational scale for the merged network [3]. In other words, the merged network can be significantly smaller, often a size similar to the largest of the individual parent networks, and perform better on a broader selection of input data than a simple combination of the original networks. In effect, the merged network learns a more optimal solution that combines elements of each of the individual parent deep networks. If you are interested in applying these techniques to your own deep-learning problems, we encourage you to contact us and explore the potential for collaboration.

ACKNOWLEDGMENTS

This research was funded under the Science Foundation Ireland (SFI) Strategic Partnership Program by SFI and FotoNation Ltd., project 13/SPP/I2868 on “Next Generation Imaging for Smartphone and Embedded Platforms.” We gratefully acknowledge the support of NVIDIA Corp. with the donation of a Titan X GPU used for this research. Portions of the research in this article use the CASIA-IrisV4 collected by the Chinese Academy of Sciences’ Institute of Automation.

ABOUT THE AUTHORS

Shabab Bazrafkan (s.bazrafkan1@nuigalway.ie) earned his B.Sc. degree from Urmia University, Iran, in electrical engineering in 2011 and his M.Sc. degree from the Shiraz University of Technology, Iran, in telecommunications engineering, image-processing branch, in 2013. Currently, he is a Ph.D. student at the Center for Cognitive, Connected, and Computational Imaging, College of Engineering and Informatics, National University of Ireland, Galway, and also works with FotoNation Ltd. His research interests are in the fields of deep neural networks and neural network design.

Peter Corcoran (dr.peter.corcoran@ieee.org) has been a university professor for 28 years and was vice dean of research

and graduate studies for seven years in the College of Engineering and Informatics, National University of Ireland, Galway. He is the co-inventor of more than 300 granted U.S. patents and has been a member of the IEEE Consumer Electronics Society for more than 20 years. He is the emeritus editor-in-chief and founding editor of *IEEE Consumer Electronics Magazine*, an industry consultant and expert witness, and the cofounder of several startup companies, including FotoNation Ltd. He is a Fellow of the IEEE.

REFERENCES

- [1] V. Vinge, “Signs of the singularity,” *IEEE Spectr.*, vol. 45, no. 6, pp. 76–82, 2008.
- [2] J. Lemley, S. Bazrafkan, and P. Corcoran, “Deep learning for consumer devices and services: Pushing the limits for machine learning, artificial intelligence, and computer vision,” *IEEE Consum. Electron. Mag.*, vol. 6, no. 2, pp. 48–56, 2017.
- [3] S. Bazrafkan, H. Javidnia, J. Lemley, and P. Corcoran. (2017). Depth from monocular images using a semi-parallel deep neural network (SPDNN) hybrid architecture. arXiv. [Online]. Available: <https://arxiv.org/abs/0811.1674v2>
- [4] S. Thavalengal and P. Corcoran, “Iris recognition on consumer devices: Challenges and progress,” in *Proc. IEEE Int. Symp. Technology and Society (ISTAS)*, 2015, pp. 1–4.
- [5] S. Thavalengal, P. Bigioi, and P. Corcoran, “Iris authentication in handheld devices: Considerations for constraint-free acquisition,” *IEEE Trans. Consum. Electron.*, vol. 61, no. 2, pp. 245–253, May 2015.
- [6] S. Thavalengal, I. Andorko, A. Drimbarean, P. Bigioi, and P. Corcoran, “Proof-of-concept and evaluation of a dual function visible/NIR camera for iris authentication in smartphones,” *IEEE Trans. Consum. Electron.*, vol. 61, no. 2, pp. 137–143, May 2015.
- [7] S. Thavalengal, P. Bigioi, and P. Corcoran, “Evaluation of combined visible/NIR camera for iris authentication on smartphones,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2015, pp. 42–49.
- [8] E. Wambui, “Samsung drives multiple modes of mobile biometrics but suffers recall setback,” *Biometric Technol. Today*, vol. 2016, no. 10, pp. 1–2, 2016.
- [9] C. Rathgeb, A. Uhl, and P. Wild, *Iris Biometrics: From Segmentation to Template Security*. New York: Springer, 2013.
- [10] J. R. Matey, R. Broussard, and L. Kennell, “Iris image segmentation and sub-optimal images,” *Image Vis. Comput.*, vol. 28, no. 2, pp. 215–222, Feb. 2010.
- [11] S. Thavalengal, P. Bigioi, and P. Corcoran, “Efficient segmentation for multi-frame iris acquisition on smartphones,” in *Proc. IEEE Int. Conf. Consumer Electronics (ICCE)*, 2016, pp. 202–203.
- [12] Chinese Academy of Sciences, Institute of Automation. (2010). Biometrics ideal test. [Online]. Available: <http://biometrics.idealtest.org/>
- [13] S. Bazrafkan and P. Corcoran, “Enhancing iris authentication on handheld devices using deep learning derived segmentation techniques,” in *Proc. IEEE Int. Conf. Consumer Electronics (ICCE)*, 2018.
- [14] S. Bazrafkan. (2017). DB augmentation. *GitHub*. [Online]. Available: https://github.com/C3Imaging/Deep-Learning-Techniques/tree/Iris_SegNet/DBaugmentation



Appendix B

Semi-Parallel Deep Neural Network (SPDNN) Hybrid Architecture, First Application on Depth from Monocular Camera

Journal of Electronic Imaging

JElectronicImaging.org

Semiparallel deep neural network hybrid architecture: first application on depth from monocular camera

Shabab Bazrafkan
Hossein Javidnia
Joseph Lemley
Peter Corcoran



Shabab Bazrafkan, Hossein Javidnia, Joseph Lemley, Peter Corcoran, "Semiparallel deep neural network hybrid architecture: first application on depth from monocular camera," *J. Electron. Imaging* **27**(4), 043041 (2018), doi: 10.1117/1.JEI.27.4.043041.

Semiparallel deep neural network hybrid architecture: first application on depth from monocular camera

Shabab Bazrafkan,[†] Hossein Javidnia,^{*,†} Joseph Lemley, and Peter Corcoran

National University of Ireland Galway, College of Engineering, Department of Electronic Engineering, Galway, Ireland

Abstract. Deep neural networks have been applied to a wide range of problems in recent years. Convolutional neural network is applied to the problem of determining the depth from a single camera image (monocular depth). Eight different networks are designed to perform depth estimation, each of them suitable for a feature level. Networks with different pooling sizes determine different feature levels. After designing a set of networks, these models may be combined into a single network topology using graph optimization techniques. This “semiparallel deep neural network (SPDNN)” eliminates duplicated common network layers and can be further optimized by retraining to achieve an improved model compared to the individual topologies. Four SPDNN models are trained and have been evaluated at two stages on the KITTI dataset. The ground truth images in the first part of the experiment are provided by the benchmark, and for the second part, the ground truth images are the depth map results from applying a state-of-the-art stereo matching method. The results of this evaluation demonstrate that using postprocessing techniques to refine the target of the network increases the accuracy of depth estimation on individual mono images. The second evaluation shows that using segmentation data alongside the original data as the input can improve the depth estimation results to a point where performance is comparable with stereo depth estimation. The computational time is also discussed in this study. © 2018 SPIE and IS&T [DOI: 10.1117/1.JEI.27.4.043041]

Keywords: deep neural networks; depth estimation; monocular camera; machine learning.

Paper 180344 received Apr. 18, 2018; accepted for publication Jul. 17, 2018; published online Aug. 7, 2018.

1 Introduction

Computing pixel depth values provides a basis for understanding the three-dimensional (3-D) geometrical structure of images. Having the depth and 3-D information of a scene enables users to infer and understand its semantics and geometric structure as well as enabling many applications in computer vision such as autonomous navigation,¹ 3-D geographic information systems,² object detection and tracking,³ medical imaging,⁴ advanced graphical applications,⁵ 3-D holography,⁶ 3-D television,⁷ multiview stereoscopic video compression,⁸ disparity-based segmentation,⁹ and human detection¹⁰/action recognition.^{11,12}

As has been presented in the recent research,¹³ using stereo images provides an accurate depth due to the advantage of having local correspondences; however, the processing time of these methods is still an open issue.

To solve this problem, it has been suggested to use single images to compute the depth values but extracting depth from monocular images requires extracting a large number of cues from the global and local information in the image. Using a single camera is more convenient in industrial applications. Stereo cameras require detailed calibration and many industrial use cases already employ single cameras, e.g., security monitoring, automotive and consumer vision systems, and camera infrastructure for traffic and pedestrian management in smart cities. These and other smart-vision applications can greatly benefit from accurate

monocular depth analysis. This challenge has been studied for a decade and is still an open research problem.

Recently, the idea of using neural networks (NN) to solve this problem has attracted attention. In this paper, we tackle this problem by employing a deep neural network (DNN) equipped with semantic pixelwise segmentation utilizing our recently published disparity postprocessing method.

This paper also introduces the use of semiparallel deep neural networks (SPDNN). An SPDNN is a semiparallel network topology developed using a graph theory optimization of a set of independently optimized convolutional neural networks (CNNs), each targeted at a specific aspect of the more general classification problem. In Refs. 14 and 15, the effect of an SPDNN approach on increasing convergence and improving model generalization is discussed. For the depth from monocular vision problem a fully connected topology, optimized for fine features, is combined with a series of max-pooled topologies (2×2 , 4×4 , and 8×8) each optimized for coarser image features. The optimized SPDNN topology is retrained on the full training dataset and converges to an improved set of network weights.

It is worth mentioning that this network design strategy is not limited to the “depth from monocular vision” problem, and further application examples and refinements will be developed in a series of future publications, currently in press.

1.1 Depth Map

Deriving the 3-D structure of an object from a set of two-dimensional (2-D) points is a fundamental problem in

*Address all correspondence to: Hossein Javidnia, E-mail: h.javidnia1@nuigalway.ie

[†]These authors contributed equally to this work.

computer vision. Most of these conversions from 2-D to 3-D space are based on the depth values computed for each 2-D point. In a depth map, each pixel is defined not by color, but by the distance between an object and the camera. In general, depth computation methods are divided into two categories:

1. Active methods.
2. Passive methods.

Active methods involve computing the depth in the scene by interacting with the objects and the environment. There are different types of active methods, such as light-based depth estimation, which uses the active light illumination to estimate the distance to different objects.¹⁶ Ultrasound and time-of-flight (ToF) are other examples of active methods. These methods use the known speed of the wave to measure the time an emitted pulse takes to arrive at an image sensor.¹⁷

Passive methods utilize the optical features of the captured images. These methods involve extracting the depth information by computational image processing. In the category of passive methods, there are two primary approaches (a) multiview depth estimation, such as depth from stereo, and (b) monocular depth estimation.

1.2 Stereo Vision Depth

Stereo matching algorithms can be used to compute depth information from multiple images. By using the calibration information of the cameras, the depth images can be generated. This depth information provides useful data to identify and detect objects in the scene.¹⁸

In recent years, many applications, including ToF,^{19,20} structured light,²¹ and Kinect, were introduced to calculate depth from stereo images. Stereo vision algorithms are generally divided into two categories: local and global. Local algorithms were introduced as statistical methods that use the local information around a pixel to determine the depth value of the given pixel. These kinds of methods can be used for real-time applications if they are implemented efficiently. Global algorithms try to optimize an energy function to satisfy the depth estimation problem through various optimization techniques.²²

In terms of computation, global methods are more complex than local methods, and they are usually impractical for real-time applications. Despite these drawbacks, they have the advantage of being more accurate than local methods. This advantage recently attracted considerable attention in the academic literature.^{23,24}

For example, the global stereo model proposed in Ref. 23 works by converting the image into a set of 2-D triangles with adjacent vertices. Later, the 2-D vertices are converted to a 3-D mesh by computing the disparity values. To solve the problem of depth discontinuities, a two-layer Markov random field (MRF) is employed. The layers are fused with an energy function allowing the method to handle the depth discontinuities. The method has been evaluated on the new Middlebury 3.0 benchmark,²⁴ and it was ranked the most accurate at the time of the paper's publication on the average weight on the "bad 2.0" index.

Another global stereo matching algorithm, proposed in Ref. 25, makes use of the texture and edge information of the image. The problem of large disparity differences in

small patches of nontextured regions is addressed by utilizing the color intensity. In addition, the main matching cost function produced by a CNN is augmented using the same color-based cost. The final results are postprocessed using a 5×5 median filter and a bilateral filter. This adaptive smoothness filtering technique is the primary reason for the algorithm's excellent performance and placement in the top of the Middlebury 3.0 benchmark.²⁴

Many other methods have been proposed for stereo depth, such as PMSC,²⁴ GCSVR,²⁴ INTS,²⁶ MDP,²⁷ and ICSG,²⁸ which all aimed to improve the accuracy of the depth estimated from stereo vision or to introduce a new method to estimate the depth from a stereo pair. However, there is always a trade-off between accuracy and speed for stereo vision algorithms.

Table 1 shows an overview of the average normalized time by the number of pixels (s/megapixels) of the most accurate stereo matching algorithms as they are ranked by the Middlebury 3.0 benchmark, based on the "bad 2.0" metric. The ranking is on the test dense set. This comparison illustrates that obtaining an accurate depth from a stereo pair requires significant processing power. These results demonstrate that today, these methods are too resource intensive for real-time applications such as street sensing or autonomous navigation due to their demand for processing resources.

To decrease the processing power of stereo matching algorithms, researchers recently began to work on depth from monocular images. Such algorithms estimate depth from a single camera while keeping the processing power low.

1.3 Deep Learning

DNN are among the most recent approaches in pattern recognition science that are able to handle highly nonlinear problems in classification and regression. These models use consecutive nonlinear signal processing units in order to mix and reorient their input data to give the most representative results. The DNN structure learns from the input and then it generalizes what it learns into data samples it has never seen before.³³ The typical DNN model is composed of one or more convolutional, pooling, and fully connected layers accompanied by different regularization tasks. Each of these units is as follows:

1.3.1 Convolutional layer

This layer typically convolves the 3-D image I with the four-dimensional kernel W and adds a 3-D bias term b to it. The output is given as

$$P = I * W + b, \quad (1)$$

where the $*$ operator is nD convolution and P is the output of the convolution. During the training process, the kernel and bias parameters are updated in a way that optimizes the error function of the network output.

1.3.2 Pooling layer

The pooling layer applies a (usually) nonlinear transform (note that the average pooling is a linear transform, but the more popular max-pooling operation is nonlinear) on

Table 1 Comparison of the performance time between the most accurate stereo matching algorithms.

Algorithm	Time/MP (s)	$W \times H$ (ndisp)	Programming platform	Hardware
PMSC ²⁴	453	1500 × 1000 (< = 400)	C++	i7-6700K, 4 GHz-GTX TITAN X
MeshStereoExt ²³	121	1500 × 1000 (< = 400)	C, C++	8 Cores-NVIDIA TITAN X
APAP-Stereo ²⁴	97.2	1500 × 1000 (< = 400)	Matlab+Mex	i7 Core 3.5 GHz, 4 Cores
NTDE ²⁵	114	1500 × 1000 (< = 400)	n/a	i7 Core, 2.2 GHz-Geforce GTX TITAN X
MC-CNN-acrt ²⁹	112	1500 × 1000 (< = 400)	n/a	NVIDIA GTX TITAN Black
MC-CNN+RBS ³⁰	140	1500 × 1000 (< = 400)	C++	Intel(R) Xeon(R) CPU E5-1650 0, 3.20 GHz, 6 Cores-32 GB RAM-NVIDIA GTX TITAN X
SNP-RSM ²⁴	258	1500 × 1000 (< = 400)	Matlab	i5, 4590 CPU, 3.3 GHz
MCCNN_Layout ²⁴	262	1500 × 1000 (< = 400)	Matlab	i7 Core, 3.5 GHz
MC-CNN-fst ²⁹	1.26	1500 × 1000 (< = 400)	n/a	NVIDIA GTX TITAN X
LPU ²⁴	3523	1500 × 1000 (< = 400)	Matlab	Core i5, 4 Cores- 2xGTX 970
MDP ²⁷	58.5	1500 × 1000 (< = 400)	n/a	4 i7 Cores, 3.4 GHz
MeshStereo ²³	54	1500 × 1000 (< = 400)	C++	i7-2600, 3.40 GHz, 8 Cores
SOU4P-net ²⁴	678	1500 × 1000 (< = 400)	n/a	i7 Core, 3.2 GHz-GTX 980
INTS ²⁶	127	1500 × 1000 (< = 400)	C, C++	i7 Core, 3.2 GHz
GCSVR ²⁴	4731	1500 × 1000 (< = 400)	C++	i7 Core, 2.8 GHz-Nvidia GTX 660Ti
JMR ²⁴	11.1	1500 × 1000 (< = 400)	C++	Core i7, 3.6 GHz-GTX 980
LCU ²⁴	9572	750 × 500 (< = 200)	Matlab, C++	1 Core Xeon CPU, E5-2690, 3.00 GHz
TMAP ³¹	1796	1500 × 1000 (< = 400)	Matlab	i7 Core, 2.7 GHz
SPS ²⁴	49.4	3000 × 2000 (< = 800)	C, C++	1 i7 Core, 2.8 GHz
IDR ³²	0.36	1500 × 1000 (< = 400)	CUDA C++	NVIDIA GeForce TITAN Black

the input image, which reduces the spatial size of the data representation after the operation.

It is common to put a pooling layer after each convolutional layer. Reducing the spatial size leads to less computational load and also prevents overfitting. The reduced spatial size also provides a certain amount of translation invariance.

1.3.3 Fully connected layer

Fully connected layers are the same as classical NN layers, where all the neurons in a layer are connected to all the neurons in their subsequent layer. The neurons give the summation of their input, multiplied by their weights, passed through their activation functions.

1.3.4 Regularization

Regularization is often used to prevent overfitting of an NN. One can train a more complex network (more parameters) with regularization and prevent overfitting. Different kinds

of regularization methods have been proposed. The most important ones are weight regularization, drop-out,³⁴ and batch normalization.³⁵ Each regularization technique is suitable for specific applications, and no single technique works for every task.

1.4 Monocular Vision Depth

Depth estimation from a single image is a fundamental problem in computer vision and has potential applications in robotics, scene understanding, 3-D reconstruction, and medical imaging.³⁶⁻³⁸ This problem remains challenging because there are no reliable cues for inferring depth from a single image. For example, temporal information and stereo correspondences are missing from such images.

As the result of the recent research, deep CNNs are setting new records for various vision applications. A deep convolutional neural field model for estimating depths from a single image has been presented in Ref. 39 by reformulating the depth estimation into a continuous conditional random

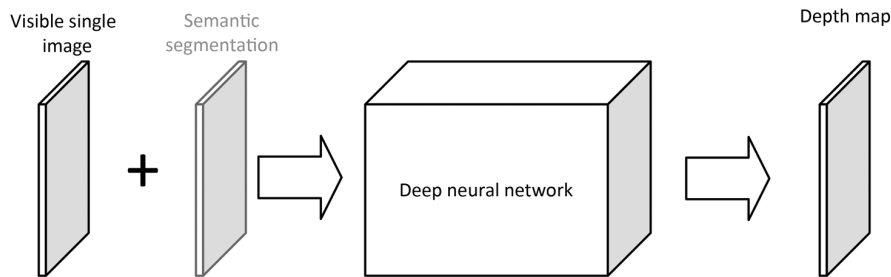


Fig. 1 The overview of the trained models in this paper. The semantic segmentation is just used in two experiments. Detailed explanation on each experiment is given in Sec. 2.3.

field (CRF) learning problem. The CNN employed in this research was composed of five convolutional and four fully connected layers. At the first stage of the algorithm, the input image was oversegmented into superpixels. The cropped image patch centered on its centroid was used as input to the CNN. For a pair of neighboring superpixels, a number of similarities were considered and were used as the input to the fully connected layer. The output of these two parts was then used as input to the CRF loss layer. As a result, the time required for estimating the depth from a single image using the trained model decreased to 1.1 s on a desktop PC equipped with NVIDIA GTX 780 GPU with 6-GB memory.

It has been found that the superpixeling technique of Ref. 39 is not a good choice to initialize the disparity estimation from mono images because of the lack of the monocular visual cues such as texture variations and gradients, defocus or color/haze in some parts of the image. To solve this issue, an MRF learning algorithm has been implemented to capture some of these monocular cues.⁴⁰ The captured cues were integrated with a stereo system to obtain better depth estimation than with the stereo system alone. This method uses a fusion of stereo + mono depth estimation.

At small distances, the algorithm relies more on stereo vision, which is more accurate than monocular vision. However, at further distances, the performance of stereo degrades; and the algorithm relies more on monocular vision.

The problem of depth estimation from monocular images has been also studied in Ref. 41, where a network is designed with two components. First, the global structure of the scene is estimated and later refined using local information. Although this approach enables the early idea of estimating monocular depth using CNNs, the output depth maps do not clearly represent the geometrical structure of the scene.

In another approach,⁴² an unsupervised convolutional encoder is trained to estimate the depth from monocular images. The depth is estimated considering the small motion between two images (stereo set as input and target). Later, the inverse warp of the target image is generated using the predicted depth and the known displacement between cameras, which results in reconstructing the source image. In a similar research,⁴³ an unsupervised CNN is trained by exploiting epipolar geometry constraints to estimate disparity from single images. The idea is to learn a function that is able to reconstruct one image from the other by utilizing a calibrated pair of binocular cameras. A left-right disparity consistency loss is also introduced, which combines smoothness, reconstruction, and left-right disparity consistency

terms and keeps the consistency between the disparities produced relative to both the left and right images.

In Refs. 44 and 45, authors presented a method to mix the output information of multiple CNNs using CRF where two different models are proposed, one with a cascade of CRFs and the other with a unified graph. They trained and tested the networks on NYU Depth V2,⁴⁶ KITTI,⁴⁷ and Make 3-D datasets.^{48,49} The best results were drawn from ResNet50.

1.5 Paper Overview

In this paper, a DNN is presented to estimate depth from monocular cameras. The depth map from the stereo sets is estimated using the same approach as Ref. 50 and they are used as the target to train the network while using information from a single image (the left image in the stereo set) as input. Four models are trained and evaluated to estimate the depth from single camera images. The network structure for all the models is the same. In the first case, the input is simply the original image. In the second case, the first channel is the original image and the second channel is its segmentation map. For each of these two cases, one of two different targets is used; specifically, these targets were the stereo depth maps with or without postprocessing explained in Ref. 50. Figure 1 shows the overview of the general approach used in this paper. In this figure, the DNN is shown as a black box. The semantic segmentation has been used in two experiments out of four. A detailed explanation of each experiment is given in Sec. 2.3.

1.6 Contributions

In this paper, two major contributions are presented:

1. SPDNN¹⁵ is a method to mix and merge several DNNs. This method is versatile enough to be applied to any DNN design. In this work, this method is utilized to design a network to approximate the depth from the monocular images. Network design procedure is described in detail in Appendix A.
2. The application of DNNs and SPDNN on estimating depth from a monocular camera.
3. The effect of using segmentation information in approximating depth is investigated.
4. Two different ground truth sets have been used to train the network and comparisons of the network performances for each ground truth have been investigated.

The rest of the paper is organized as follows: in the next section, the network structure, database preparation, and the

training process are presented. Section 3 discusses the results and evaluation of the proposed method. The conclusion and discussions are presented in the last section.

2 Methodology

2.1 Network Structure

2.1.1 Semiparallel deep neural network

This paper introduces the SPDNN concept, inspired by graph optimization techniques. In this method, several DNNs are parallelized and merged in a way that facilitates the advantages of each. The final model is trained for the problem. References 14 and 15 show that using this method increases the convergence and generalization of the model compared to alternatives.

The merging of multiple networks using SPDNN is described in the context of the current depth mapping problem. In this particular problem, eight different networks were designed for the depth estimation task. These are described in detail in Appendix A. None of these networks on their own gave useful results on the depth analysis problem. However, it was noticed that each network tended to perform well on certain aspects of this task while failing at others. This led to the idea that it would be advantageous to combine multiple individual networks and train them in a parallelized architecture. Our experiments showed that better output could be achieved by merging the networks and then training them concurrently.

Combined model/architecture. The process of the network design is discussed in detail in Appendix A. In the final model presented in Fig. 2, the input image is first processed in four parallel fully convolutional subnetworks with different pooling sizes. This provides the advantages of different networks with different pooling sizes at the same time. The outputs of these four subnetworks are concatenated in two different forms; one is to pool the larger images to be the same size as the smallest image in the previous part, and the other one is to unpool the smaller images of the previous part to be the same size as the largest image.

After merging these outputs, the data are led to two different networks. One is the fully convolutional network (FCN)

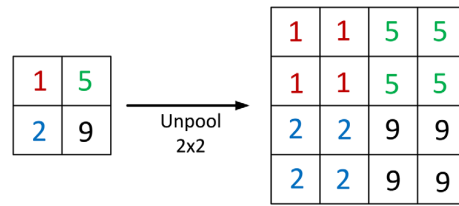


Fig. 3 The repeating technique used in unpooling layers.

to deepen the learning and release more abstract features of the input, and the other network is an autoencoder network with different architectures for encoder and decoder.

It is mentioned in the network design section in Appendix A that having a fully connected layer in the network is crucial for correct estimation of the image's depth, which is provided in the bottleneck of the autoencoder. The results from the autoencoder and the fully convolutional subnetwork are again merged in order to give a single output after applying a one channel convolutional layer.

In order to regularize the network, prevent overfitting, and increase the convergence, batch normalization³⁵ is applied after every convolutional layer, and the drop-out technique³⁴ is used in fully connected layers. The experiments in this paper show that using weight regularization in the fully connected layers gives slower convergence; therefore, this regularization was eliminated from the final design. All the nonlinearities in the network are the ReLU nonlinearity, which is widely used in DNNs, except for the output layer, which took advantage of the sigmoid nonlinearity. The value repeating technique was used in the unpooling layer due to nonspecificity of the corresponding pooled layer in the decoder part of the autoencoder subnetwork.

The value repeating technique, shown in Fig. 3, involves repeating the value from the previous layer in order to obtain the unpooled image. The figure shows the 2×2 unpooling, and the process is the same for other unpooling sizes.

2.2 Database

In this paper, the KITTI Stereo 2012, 2015 datasets⁴⁷ are used for training and evaluation of the network. The database is augmented by vertical and horizontal flipping to expand

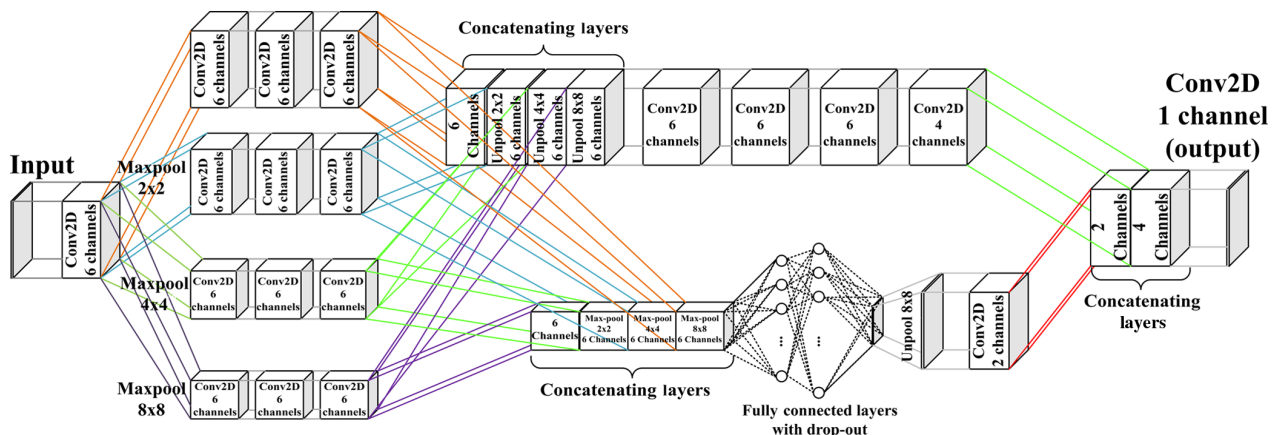


Fig. 2 The model designed for the depth estimation from monocular images. The network design is explained in Appendix A.2.

the total size to 33,096 images. 70% of this dataset is used for training, 20% for validation, and 10% for testing. Dividing the database to train-validation-test subsets is performed before scrambling the indices so there is minimum correspondence between the samples in each subset. The reason for this is because the database is drawn from sequences of images wherein each two consecutive samples look very similar. Each model is trained for two sets of input samples and two sets of output targets. The input and target preparation are explained in the following sections.

2.2.1 Data preparation

Input preparation. Two different sets have been used as the input of the network. The first set includes the visible images given by the left camera. The second set is the visible image + the semantic segmentation of the corresponding input. This gives the opportunity of investigating the segmentation influence on the depth estimation problem. The segmentation map for each image is calculated by employing the well-known model “SegNet.”^{51,52} This model is one of the most successful recent implementations of DNN for semantic pixelwise image segmentation and has surpassed other configurations of FCNs both in accuracy and simplicity of implementation. A short description of SegNet is given in Appendix B.

In our experiments, SegNet was trained using stochastic gradient descent with learning rate 0.1 and momentum 0.9. In this paper, the Caffe implementation of SegNet has been employed for training purposes.⁵³ The gray-scale CamVid road scene database (360×480)⁵⁴ has been used in the training step.

Target preparation. The targets for training the network are generated from the stereo information using the adaptive random walk with restart algorithm.⁵⁵ The output of the stereo matching algorithm suffers from several artifacts, which are addressed and solved by a postprocessing method in Ref. 50. In the present experiments, both depth maps (before postprocessing and after postprocessing) are used independently as targets. The postprocessing procedure is based on the mutual information of the RGB image (used as a reference image) and the initial estimated depth image. This approach has been used to increase the accuracy of the depth estimation in stereo vision by preserving the edges and corners in the depth map and filling in the missing parts. The method was compared with the top eight depth estimation methods in the Middlebury benchmark²⁴ at the time the paper was authored. Seven metrics, including mean square error (MSE), root mean square error (RMSE), peak signal-to-noise ratio (PSNR), signal-to-noise ratio (SNR), mean absolute error (MAE), structural similarity index (SSIM), and structural dissimilarity index were used to evaluate the performance of each method. The evaluation ranked the method as first in five metrics and second and third in other metrics

2.3 Training

As described in Sec. 2.2.1, there are two separate sets as inputs and two separate sets as targets for the training process. This will give four experiments in total as follows:

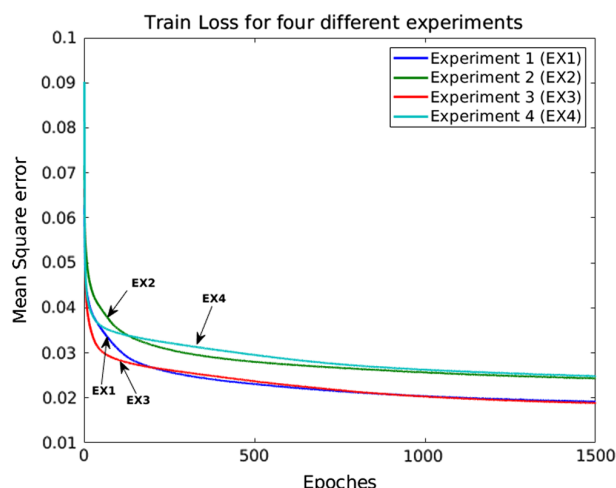


Fig. 4 Train loss for each experiment.

1. Experiment 1: Input: left visible image + pixelwise segmented image. Target: postprocessed depth map
2. Experiment 2: Input: left visible image. Target: postprocessed depth map.
3. Experiment 3: Input: left visible image + pixelwise segmented image. Target: depth map.
4. Experiment 4: Input: left visible image. Target: depth map.

The images are resized to 80×264 pixels during the whole process. Training is done on a standard desktop with an NVIDIA GTX 1080 GPU with 8 GB memory.

In the presented experiments, the MSE value between the output of the network and the target values have been used as the loss function, and the Nesterov momentum technique⁵⁶ with learning rate 0.01 and momentum 0.9 has been used to train the network. The training and validation loss for each of these experiments are shown in Figs. 4 and 5, respectively.

These figures show that using the postprocessed depth map as the target results in lower loss values, which means that the network was able to learn better features in those

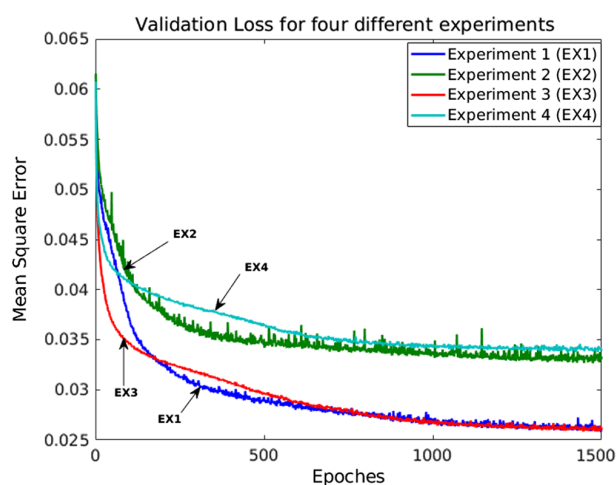


Fig. 5 Validation loss for each experiment.

experiments, while semantic segmentation decreases the error only marginally.

The reason that the postprocessed depth maps are considered as the target in two experiments is twofold: first, the post-processing pipeline is proven to be effective in increasing the performance of the depth estimation methods by considering the geometrical structure of the scene. Second, it helps the network to avoid the densification process of the sparse depth maps, which are captured using LIDAR scanners.

3 Results and Evaluations

The evaluation in this paper has been done in four parts. In the first two parts, the four experiments given in Sec. 2.3 are compared to each other, given different ground truths. The third part compares the proposed method to a stereo matching method and the last part shows the comparison against the state-of-the-art monocular depth estimation method. For evaluation purposes, eight metrics including PSNR, MSE (between 0 and 1), RMSE (between 0 and 1), SNR, MAE (between 0 and 1), SSIM (between 0 and 1),⁵⁷ universal quality index (UQI) (between 0 and 1),⁵⁸ and Pearson correlation coefficient (PCC) (between -1 and 1)⁵⁹ are used. For the metrics PSNR, SNR, SSIM, UQI, and PCC, the larger value indicates better performance, and for MSE, RMSE, and MAE, the lower value indicates better performance. PSNR, MSE, RMSE, MAE, and SNR represent the general similarities between two objects. UQI and SSIM are structural similarity indicators and PCC represents the correlation between two samples. To the best of our knowledge, there have been no other attempts at estimating depth from a mono camera on the KITTI benchmark.

3.1 Comparing Experiments Given Benchmark Ground Truth

The KITTI database came with a depth map ground truth generated by a LIDAR scanner.

The test set has been forward propagated through the four different models trained in the four experiments, and the output of the networks has been compared to the benchmark ground truth. The results are shown in Table 2. The best value for each metric is presented in bold.

Table 2 Numerical comparison of the models given the benchmark's ground truth.

	Exp. 1	Exp. 2	Exp. 3	Exp. 4
PSNR	14.3424	13.7677	13.8333	13.8179
MSE	0.0382	0.0436	0.0435	0.0439
RMSE	0.1937	0.2069	0.206	0.2066
SNR	4.4026	3.8279	6.1952	6.1798
MAE	0.1107	0.1212	0.1236	0.1234
SSIM	0.9959	0.9955	0.9955	0.9955
UQI	0.9234	0.9252	0.9053	0.9064
PCC	0.7687	0.8485	0.7702	0.7729

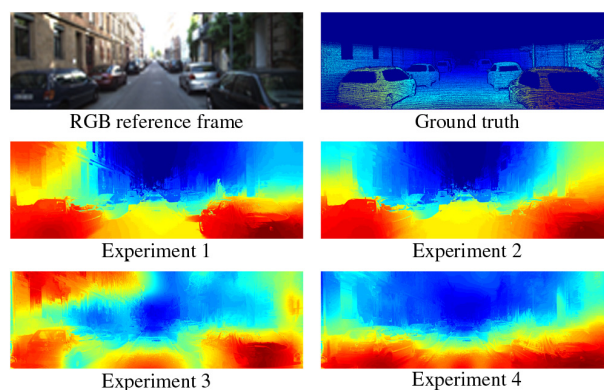


Fig. 6 Estimated depth maps from the trained models, example 1, for experiments 1 to 4 explained in Sec. 2.3.

Figures 6–8 represent the color-coded depth maps computed by the trained models using the proposed DNN, where the dark red and dark blue parts represent closest and furthest points to the camera, respectively. On the top right of each figure, the ground truth given by the benchmark is illustrated. For visualization purposes, all of the images presented in this section are upsampled using joint bilateral upsampling.⁶⁰ The results show that using semantic segmentation along with the visible image as input will improve the model

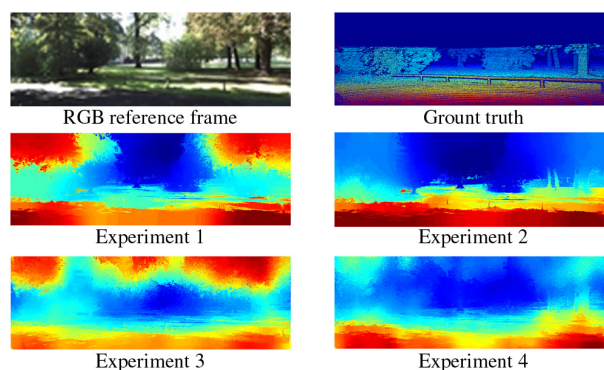


Fig. 7 Estimated depth maps from the trained models, example 2, for experiments 1 to 4 explained in Sec. 2.3.

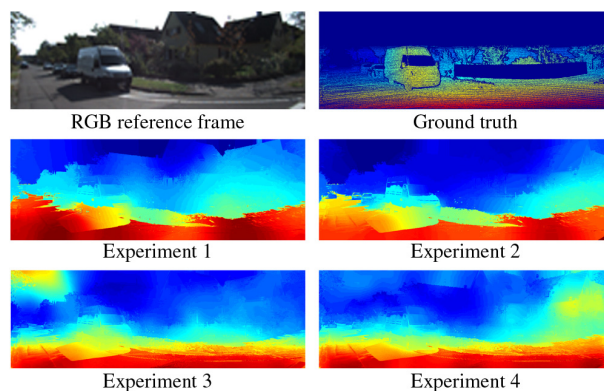


Fig. 8 Estimated depth maps from the trained models, example 3, for experiments 1 to 4 explained in Sec. 2.3.

marginally. Using the postprocessed target in the training stage helps the model to converge to more realistic results.

As it is shown in Figs. 6–8, the depth map generated in experiment 1 contains more structural details, and more precise, less faulty depth levels compared with the other experiments. In general, the presented models in this paper are able to handle occlusions and discontinuities at different depth levels.

3.2 Comparing Experiments Given the Ground Truth from Stereo Matching

In this section, proposed models are compared to see which one produces closer results to the target value. This gives an idea whether using deep learning techniques on the mono camera can produce reasonable results or not.

Images in the test set have been forward propagated through the models trained in Sec. 2.3, and the outputs are compared with the depth map generated in Ref. 50. The numerical results are shown in Table 3.

The best value for each metric is presented in bold. Figures 9–11 represent the color-coded depth maps computed by the trained models using the proposed DNN,

Table 3 Numerical comparison of the models given the ground truth from stereo matching.

	Exp. 1	Exp. 2	Exp. 3	Exp. 4
PSNR	15.0418	14.1895	13.3819	14.0491
MSE	0.0378	0.0447	0.0535	0.0441
RMSE	0.1854	0.203	0.2223	0.2039
SNR	8.822	7.9696	5.4271	6.0943
MAE	0.1442	0.1581	0.1673	0.153
SSIM	0.9952	0.9943	0.994	0.9951
UQI	0.8401	0.8369	0.7951	0.8178
PCC	0.8082	0.795	0.704	0.6919

where the dark red and dark blue parts represent closest and furthest points to the camera, respectively. On the top right of each figure, the ground truth calculated in Ref. 50 is illustrated. For visualization purposes, all of the images presented in this section are upsampled using joint bilateral upsampling.⁶⁰

The results show that using semantic segmentation along with the visible image as input will improve the model marginally. Using the postprocessed target in the training stage helps the model to converge to more realistic results.

Figures 9–11 show that the trained models in this paper are able to estimate depth maps comparable to state-of-the-art stereo matching with structural accuracy and precise depth levels. This is also a result of using the semantic segmentation data and injecting the structural information into the network.

3.3 Comparing Mono Camera Results with Stereo Matching

In this section, the results from the mono camera depth estimation given by the proposed method are compared with one of the top-ranked stereo matching methods given in Ref. 50. The ground truth for this comparison is the set of depth maps provided by the KITTI benchmark.

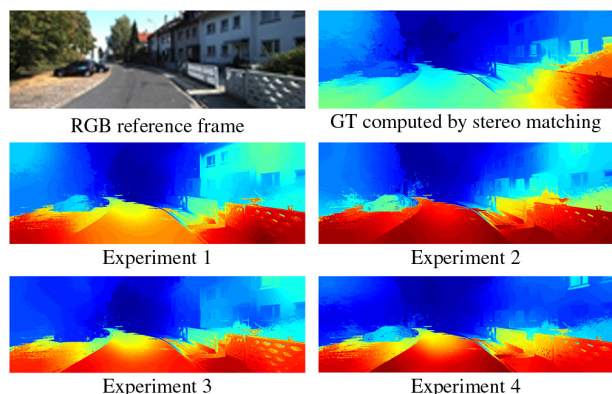


Fig. 9 Estimated depth maps from the trained models, example 1, for experiments 1 to 4 explained in Sec. 2.3.

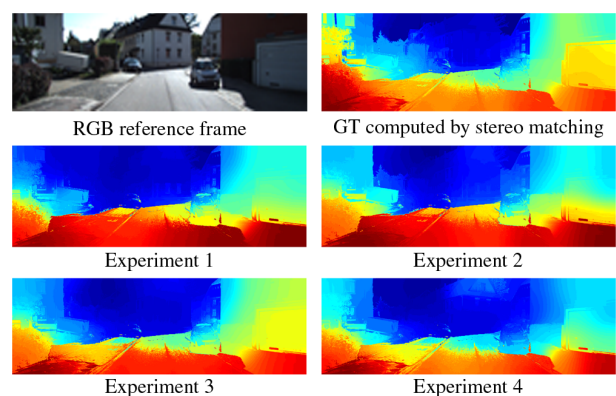


Fig. 10 Estimated depth maps from the trained models, example 2, for experiments 1 to 4 explained in Sec. 2.3.

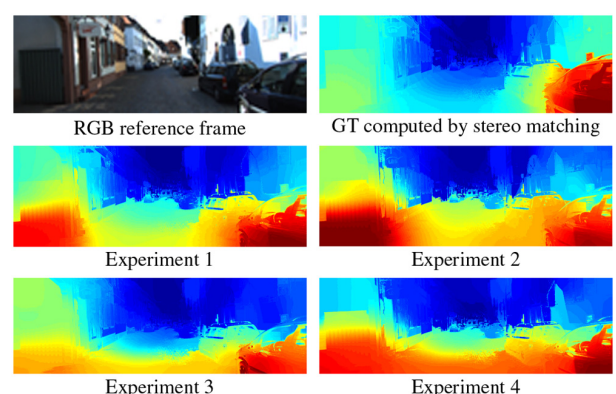


Fig. 11 Estimated depth maps from the trained models, example 3, for experiments 1 to 4 explained in Sec. 2.3.

Table 4 Numerical comparison between stereo matching and the proposed mono camera model.

	Stereo matching ⁵⁰	Mono camera DNN
PSNR	14.8234	14.3424
MSE	0.0351	0.0382
RMSE	0.1845	0.1937
SNR	4.8836	4.4026
MAE	0.1017	0.1107
SSIM	0.9966	0.9959
UQI	0.9353	0.9234
PCC	0.823	0.7687

The test images have been forward propagated through the models trained in Sec. 2.3 and the best results are compared with the stereo matching technique. The results are shown in Table 4.

The results indicate that using mono camera images and deep learning techniques can provide results that are comparable to stereo matching techniques. As shown in Table 4, the mono camera DNN method was able to provide depth maps similar to the stereo matching methods, represented by PSNR, MSE, MAE, RMSE, and SNR.

Having close values for SSIM (0.9966 and 0.9959 in the range [0,1]) and UQI (0.9353 and 0.9234 in the range [0,1]) shows how the mono camera DNN method is able to preserve the structural information, as compared to the stereo matching method.

3.4 Comparison against Other Monocular Depth Estimation Methods

In this section, the proposed network is compared against the method presented in Refs. 39 and 41–43. Table 5 represents the performance of the proposed network compared to the state-of-the-art methods based on seven metrics including absolute relative difference, squared relative difference, and RMSE/RMSE log. The metrics are defined as follows:

- Mean relative error (Rel): $\frac{1}{P} \sum_{i=1}^P \frac{|\tilde{d}_i - d_i^*|}{d_i^*}$;
- Root mean squared error (RMSE): $\sqrt{\frac{1}{P} \sum_{i=1}^P (\tilde{d}_i - d_i^*)^2}$;
- Mean log 10 error: $\frac{1}{P} \sum_{i=1}^P \|\log_{10}(\tilde{d}_i) - \log_{10}(d_i^*)\|$;
- Accuracy with threshold t : Percentage (%) of d_i^* , subject to $\max\left(\frac{d_i^*}{\tilde{d}_i}, \frac{\tilde{d}_i}{d_i^*}\right) = \delta < t$ ($t \in [1.25, 1.25^2, 1.25^3]$).

These numbers indicate that the unsupervised CNN proposed by Godard et al.⁴³ outperforms the others because of the left-right disparity consistency term, which allows the network to optimize the disparity values based on both left and right images. However, we believe that the proposed network has a competitive performance compared to the studied methods considering the fact that our models are trained using only the left image without taking into account the influence of the right disparity values.

3.5 Comparing Running Times

In this section, the computational time of the proposed method is compared against the stereo matching methods provided in Table 1. The evaluations indicate that the proposed method is able to perform at a rate of ~ 1.23 s/MP on a desktop computer equipped with i7 2600 CPU @ 3.4 GHz and 16 GB of RAM.

Table 5 Results on the KITTI 2015 stereo 200 training set disparity images.

Method	Stereo	Dataset	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Eigen et al. ⁴¹ coarse	No	KITTI	0.361	4.826	8.102	0.377	0.638	0.804	0.894
Eigen et al. ⁴¹ fine	No	KITTI	0.203	1.548	6.307	0.282	0.702	0.890	0.958
Liu et al. ³⁹ DCNF-FCSP FT	No	KITTI	0.201	1.584	6.471	0.273	0.68	0.898	0.967
Garg et al. ⁴² L12 Aug 8× cap 50 m	Yes	KITTI	0.169	1.080	5.104	0.273	0.740	0.904	0.962
Godard et al. ⁴³	Yes	KITTI	0.148	1.344	5.927	0.247	0.803	0.922	0.964
Saxena et al. ⁶¹	No	KITTI	0.280	—	8.734	0.327	0.601	0.820	0.926
Zhou et al. ⁶²	No	KITTI	0.208	1.768	6.858	—	0.678	0.885	0.957
Kuznetsov et al. ⁶³ (only supervised)	No	KITTI	—	—	4.815	—	0.845	0.957	0.987
Kuznetsov et al. ⁶³	Yes	KITTI	—	—	4.621	—	0.852	0.960	0.986
Xu et al. ⁴⁴	No	KITTI	0.125	0.899	4.685	0.154	0.816	0.951	0.983
Ours	No	KITTI	0.288	1.065	4.071	0.401	0.51	0.77	0.893

In columns 4–7 lower is better, in columns 8–10 higher is better.

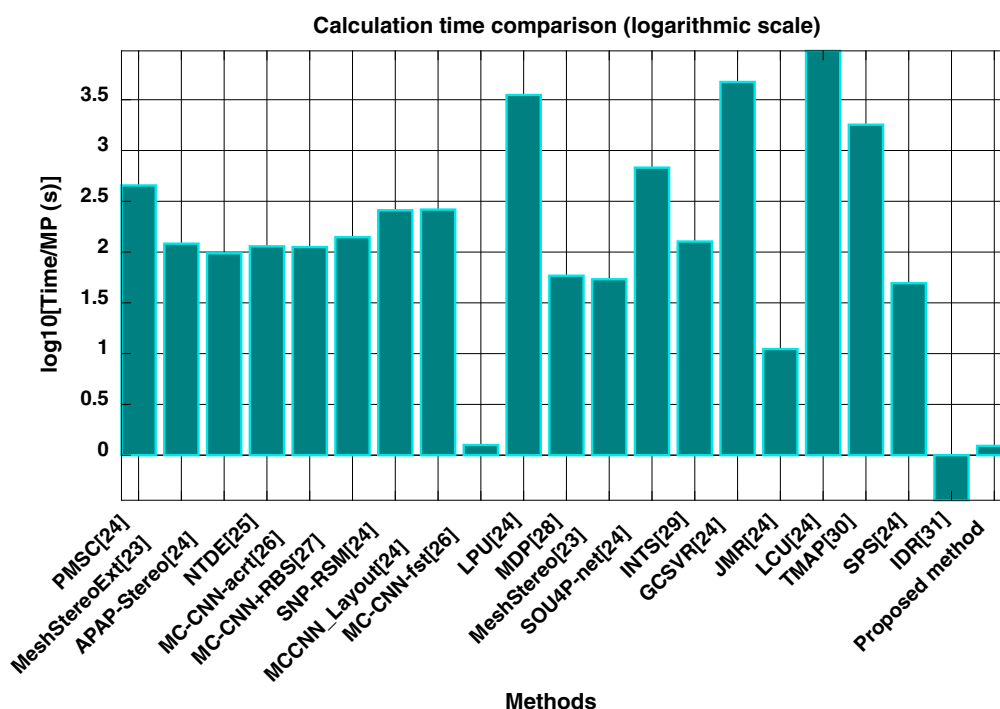


Fig. 12 Comparison of computational time in logarithmic scale.

Figure 12 shows the comparison of the computational times. The comparison is done in a logarithmic scale due to the large range of computational times between different methods.

3.6 Effects of Scaling, Rotation, and Translation

In this section, the effect of the scaling, rotation, and translation is explained for the proposed method. The test data have been manipulated in three ways:

1. Scaling: Images have been cropped with random values for height in $U[37,70]$ and width in $U[128,240]$ where $U[a, b]$ is the uniform distribution between a and b . The cropped images are resized to $[80, 264]$ using bilinear interpolation.

2. Rotating: Each sample in the test set has been rotated randomly between 3 deg and 30 deg and also -3 deg and -30 deg.
3. Translating: Each sample has been translated with random values between 5 and 30 pixels in height and 50 and 100 pixels in width.

Each set of these samples has been tested on the proposed networks for each experiment and the results are given in Tables 6–9.

This experiment shows that the method is relatively robust to scaling in comparison to rotation and translation. Translation introduces more error than rotation. The main reason is the change in the sky position in the translated images. Since the network is trained on samples where the

Table 6 Network trained in experiment 1, tested on scaled, rotated, and translated samples.

	Exp. 1, original	Exp. 1, scaled	Exp. 1, rotated	Exp. 1, translated
PSNR	14.3424	13.4155	7.3789	6.9341
MSE	0.0382	0.0561	0.1890	0.2083
RMSE	0.1937	0.2253	0.4313	0.4533
SNR	4.4026	1.4561	3.0047	2.5507
MAE	0.1107	0.1811	0.3480	0.3609
SSIM	0.9959	0.9924	0.9791	0.9754
UQI	0.9234	0.2234	0.343	0.530

Table 7 Network trained in experiment 2, tested on scaled, rotated, and translated samples.

	Exp. 2, original	Exp. 2, scaled	Exp. 2, rotated	Exp. 2, translated
PSNR	13.7677	13.1384	7.1016	6.7348
MSE	0.0436	0.0600	0.2017	0.2218
RMSE	0.2069	0.2328	0.4454	0.4660
SNR	3.8279	1.8092	3.7111	2.8531
MAE	0.1212	0.1882	0.3501	0.3665
SSIM	0.9955	0.9919	0.9776	0.9738
UQI	0.9252	0.2244	0.708	0.764

Table 8 Network trained in experiment 3, tested on scaled, rotated, and translated samples.

	Exp. 3, original	Exp. 3, scaled	Exp. 3, rotated	Exp. 3, translated
PSNR	13.8333	11.5803	7.2379	6.2700
MSE	0.0435	0.0808	0.1962	0.2420
RMSE	0.206	0.2740	0.4388	0.4890
SNR	6.1952	4.1375	4.6070	5.3933
MAE	0.1236	0.2187	0.3580	0.3787
SSIM	0.9955	0.9897	0.9780	0.9715
UQI	0.9053	0.826	0.293	0.489

Table 9 Network trained in experiment 4, tested on scaled, rotated, and translated samples.

	Exp. 4, original	Exp. 4, scaled	Exp. 4, rotated	Exp. 4, translated
PSNR	13.8179	12.0590	8.0581	7.5241
MSE	0.0439	0.0725	0.1649	0.1847
RMSE	0.2066	0.2595	0.4009	0.4253
SNR	6.1798	3.6592	3.5843	4.3851
MAE	0.1234	0.2023	0.3191	0.3294
SSIM	0.9955	0.9908	0.9817	0.978
UQI	0.9064	0.878	0.250	0.468

sky is at the top of the image, the translating the sky position induces a large amount of uncertainty on the output values.

The other observation is for using segmentation as auxiliary information for depth estimation. The observations show that the segmentation is not introducing any helpful information while dealing with scaling, rotation, and translation.

4 Conclusion and Discussion

In this paper, we have introduced the use of the SPDNN method. An SPDNN is a network topology developed using a graph theory optimization of a set of independently optimized CNNs, each targeted at a specific aspect of the more general classification problem. For depth estimation from a monocular setup, a model including fully connected topology optimized for fine features is combined with a series of max-pooled topologies. The optimized SPDNN topology is retrained on the full training dataset and converges to an improved set of network weights. Here, we used this design strategy to train an accurate model for estimating depth from monocular images.

In this work, eight different DNNs have been mixed and merged using the SPDNN method in order to take advantage

of each network's qualities. The mixed network architecture was then trained in four separate scenarios where each scenario used a different set of inputs and targets during training. Four distinct models have been trained. The pixelwise segmentation and depth estimations given in Ref. 50 were used to provide samples for use in the training stage. The KITTI benchmark was used for training and experimental purposes.

Each model was evaluated in two sections, first against the ground truth provided by the benchmark, and second against the disparity maps computed by the stereo matching method (Secs. 3.1 and 3.2). The results show that using the postprocessed depth map presented in Ref. 50 for training the network results in more precise models and adding the semantic segmentation of the input frame to the input helps the network preserve the structural information in the output depth map. The results in Sec. 3.2 show how close the proposed depth estimation using mono camera can be to the stereo matching method. The semantic segmentation information helps the network converge to the stereo matching results, although the improvement is marginal in this case. The results of the third comparisons in Sec. 3.3 show a slightly higher accuracy obtained by employing the stereo matching technique, but our results demonstrate that there is not a big difference between the depths from the models trained by the proposed DNN and the values computed by stereo matching. The numerical results of this evaluation show the similarity between the mono camera using the DNN method and the stereo matching method, and also the power of the presented method in preserving the structural information in the output depth map.

An important advantage of these models is the processing time of ~ 1.23 s/MP. This is equal to 38 fps for an input image of size (80×264) on an i7 2600 CPU @ 3.4 GHz and 16 GB of RAM. This makes the model suitable for providing depth estimation in real time. This performance is comparable to the stereo methods MC-CNN-fst²⁹ and JMR,²⁴ whose times are 37 and 4 fps, respectively, for the same size of the image, taking advantage of GPU computation power (NVIDIA GTX TITAN X and GTX 980, respectively). The IDR method³² can give up to 131 fps for the same image size by using an NVIDIA GeForce TITAN Black GPU and CUDA C++ implementation, but the performance on a CPU is not given by the authors, so any comparisons with this method would be unfair.

Using pixelwise segmentation as one of the inputs of the network slightly increased the accuracy of the models, and also helped the model preserve the structural details of the input image. However, it also brought some artifacts, such as wrong depth patches on the surfaces. The evaluation results also illustrate the higher accuracy of the models where a postprocessed depth map was used as the target in the training procedure.

4.1 Future Works and Improvements

The model presented in this work is still a big model to implement in low power consumer electronic devices (e.g., handheld devices). Future work will include a smaller design that is able to perform as well as the presented model. The other consideration for the current method is the training data size (which is always the biggest consideration with deep learning approaches). The amount of stereo data available

in the databases is usually not big enough to train a DNN. The augmentation techniques can help to expand databases, but the amount of extra information they provide is limited. Providing a larger set with accurate depth maps will improve the results significantly.

The future works also involve designing and training networks on other databases, such as NYU Depth V2⁴⁶ and Make3D,^{48,49} and performing interdatabase evaluation wherein the network is trained on one database and tested on another one. We will also utilize the SPDNN method to design a network for a mixed database to get more generalization power.

The SPDNN approach is currently being applied to other problems and is giving promising results on both classification and regression problems. Those results will be presented in future publications.

Appendix A: Network Design

A1 Individual Networks for Depth Analysis

The network shown in Fig. 13 is a deep fully CNN (a fully CNN is a network wherein all the layers are convolutional layers) with no pooling and no padding. Therefore, no information loss occurs inside the network, as there is no bottleneck or data compression; this network is able to preserve the details of the input samples. But the main problem is that this model is unable to find big objects and coarse features in the image. In order to solve this problem, three other networks have been designed as shown in Figs. 14–16. These three networks take advantage of the max-pooling layers to gain transition invariance and also to recognize larger objects and coarser features inside the image. These networks use 2×2 , 4×4 , and 8×8 max-pooling operators,

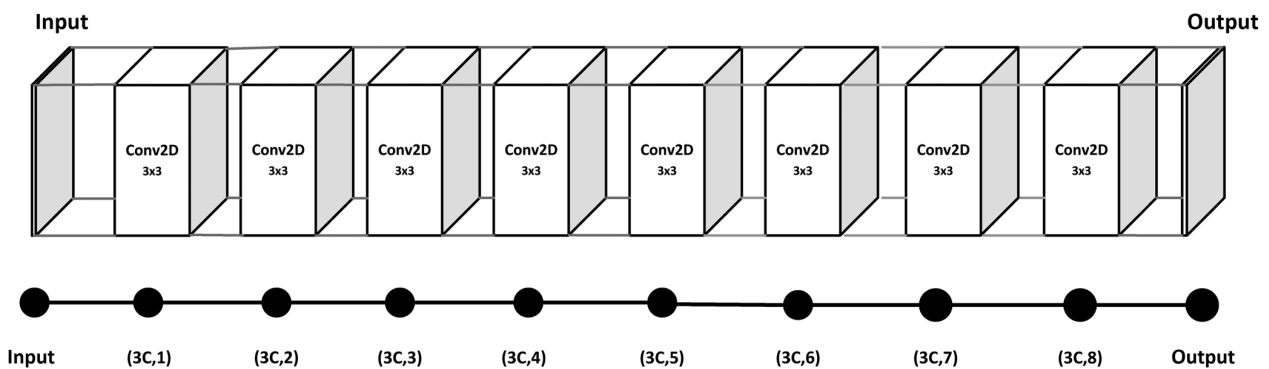


Fig. 13 Top row: network 1, Bottom row: graph corresponds to network 1.

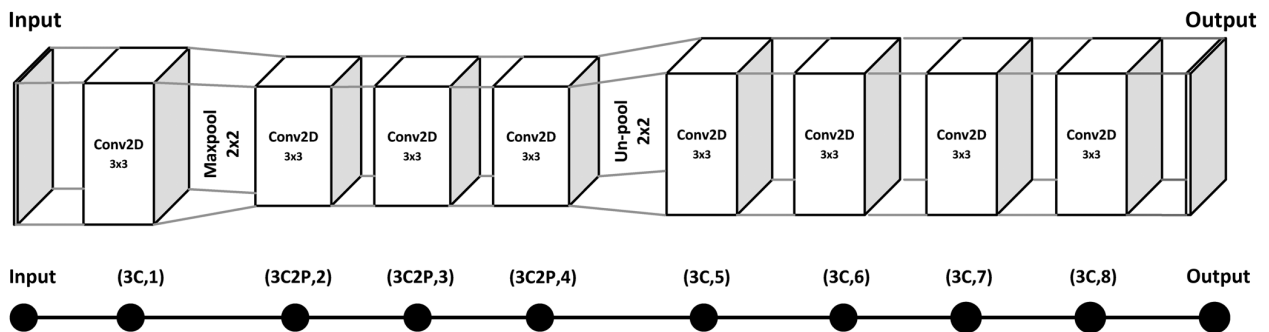


Fig. 14 Top row: network 2, Bottom row: graph corresponds to network 2.

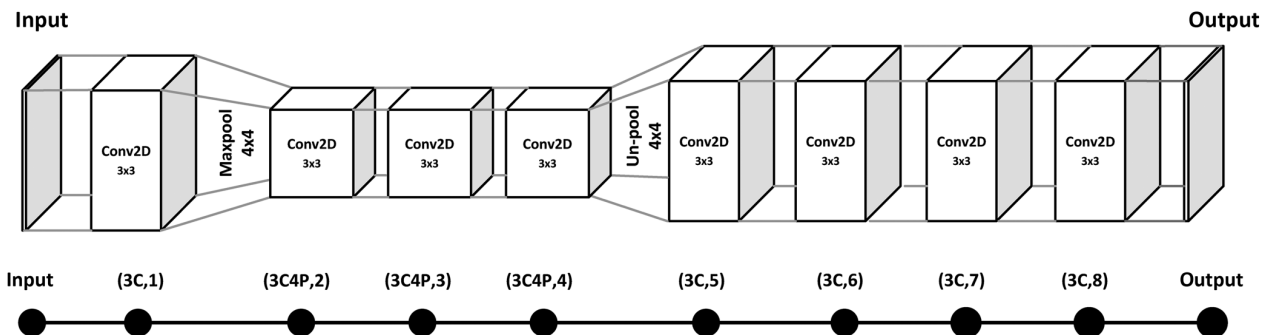


Fig. 15 Top row: network 3, Bottom row: graph corresponds to network 3.

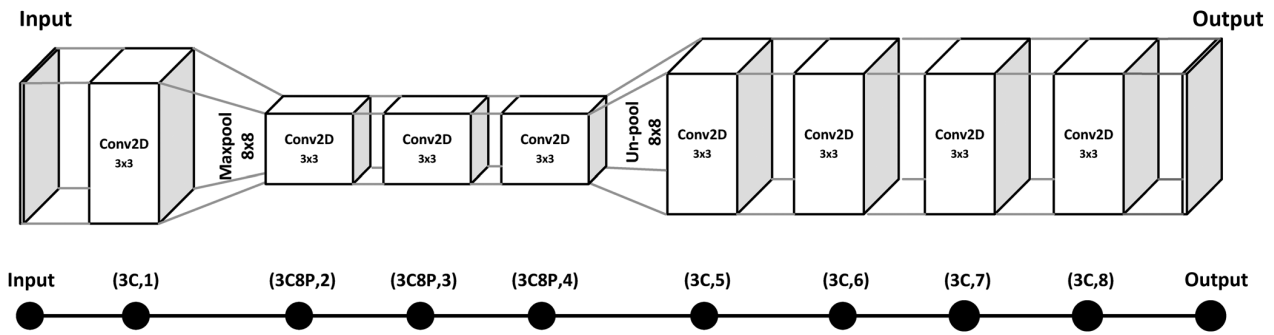


Fig. 16 Top row: network 4, Bottom row: graph corresponds to network 4.

respectively. Larger pooling kernels allow coarser features to be detected by the network. The main problem with these networks is that the spatial details vanished as a result of data compression in pooling layers.

After several attempts of designing different networks, the observations showed that in order to estimate the depth from an image, the network needed to see the whole image as one object. To do that requires the kernel to be the same size as the image in at least one layer that is equivalent to a fully connected layer inside the network.

In fully connected layers, each neuron is connected to all neurons in the previous/next layer. Due to the computationally prohibitive nature of training fully connected layers and their tendency to cause overfitting, it is desirable to

reduce the number of these connections. Adding fully connected layers results in a very tight bottleneck, which seems to be crucial for the depth estimation task, but also causes the majority of the details in the image to be lost. In Figs. 17–20, the networks with fully connected layers are shown. These networks correspond to networks in Figs. 13–16 but with convolutional layers replaced with fully connected layers on the righthand side of the network. Using different pooling sizes before the fully connected layer will cause the network to extract different levels of features, but all these configurations introduce loss of detail.

Each of these eight configurations has its own advantages and shortcomings, from missing the coarse features to

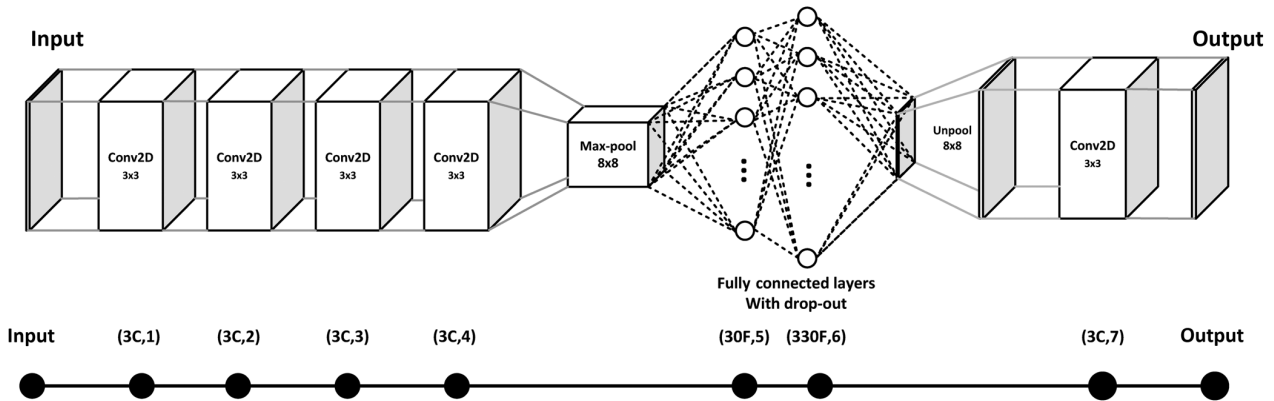


Fig. 17 Top row: network 5, Bottom row: graph corresponds to network 5.

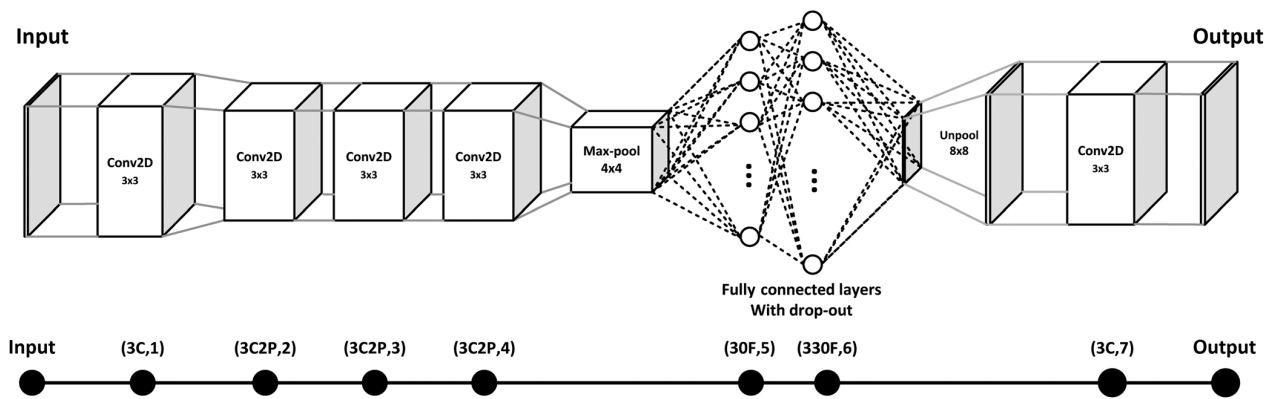


Fig. 18 Top row: network 5, Bottom row: graph corresponds to network 6.

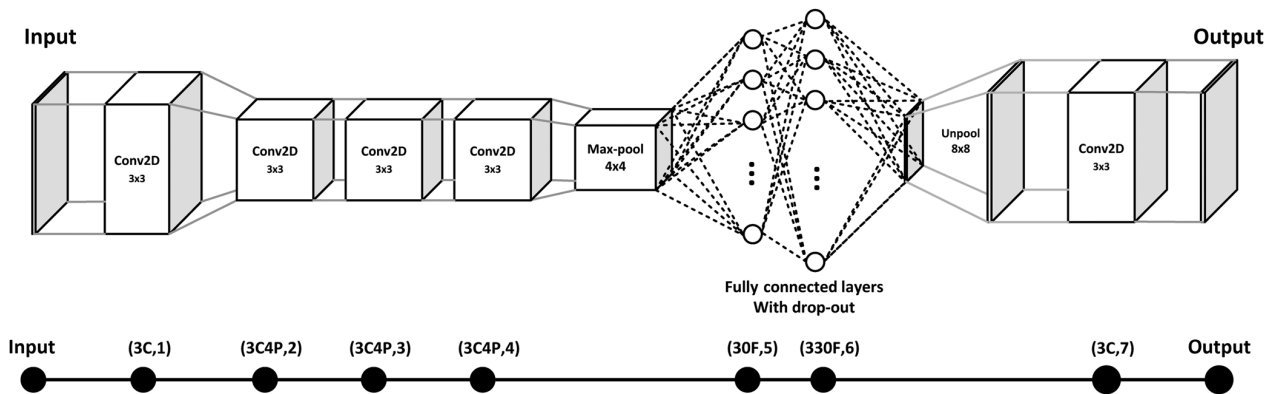


Fig. 19 Top row: network 7, Bottom row: graph corresponds to network 7.

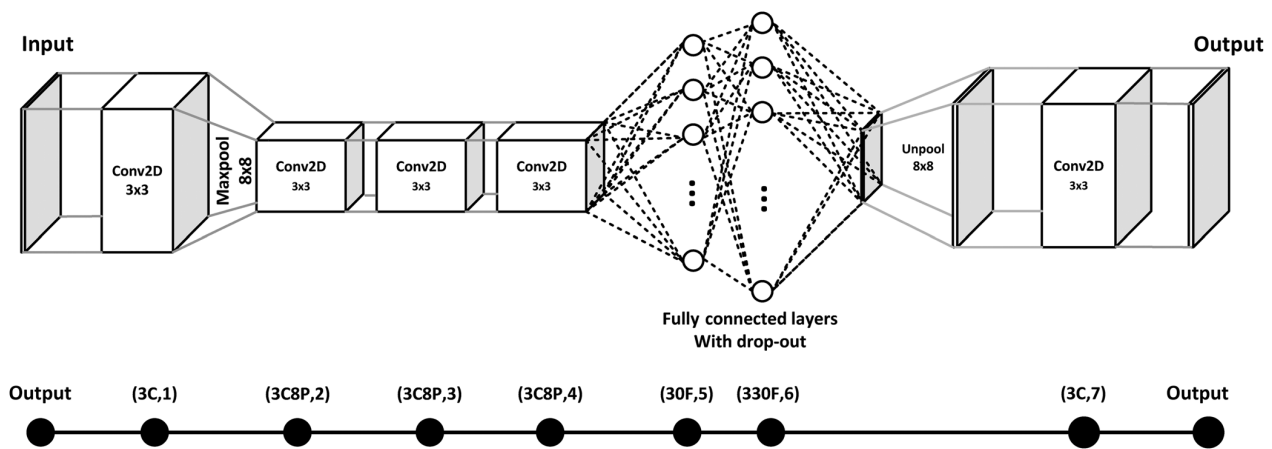


Fig. 20 Top row: network 8, Bottom row: graph corresponds to network 8.

missing the details. None of these designs converged to a reasonable depth estimation model.

The main idea of the SPDNN method is to mix and merge these networks and generate a single model, which includes all the layers of the original models in order to be able to preserve the details and also detect the bigger objects in the scene for the depth estimation task.

A.2 SPDNN Parallelization Methodology

A.2.1 Graph Contraction

A consideration while parallelizing NNs is that having the same structure of layers with the same distance from the input might lead all the layers to converge to similar values. For example, the first layer in all of the networks shown in Figs. 13–20 is a 2-D convolutional layer with a 3×3 kernel.

The SPDNN idea uses graph contraction to merge several NNs. The first step is to turn each network into a graph in which it is necessary to consider each layer of the network as a node in the graph. Each graph starts with the input node and ends with the output node. The nodes in the graph are connected based on the connections in the corresponding layer of the network. Note that the pooling and unpooling layers are not represented as nodes in the graph, but their properties will stay with the graph labels, which will be explained later.

Figures 13–20 presents the networks and their corresponding compressed graphs. Two properties are assigned to each node in the graph. The first property is the layer structure, and the second one is the distance of the current node to the input node. To convert the network into a graph, a labeling scheme is required. The proposed labeling scheme uses different signs for different layer structures, C for convolutional layer (e.g., 3C mean a convolutional layer with 3×3 kernel), F for fully connected layer (e.g., 30F means a fully connected layer with 30 neurons), and P for pooling property (e.g., 4P means that the data have been pooled by the factor of 4 in this layer).

Some properties, such as convolutional and fully connected layers, occur in a specific node, but pooling and unpooling operations will stick with the data to the next layers. The pooling property stays with the data except when an unpooling or a fully connected layer is reached. For example, a node with the label (3C8P, 4) corresponds to a convolutional layer with a 3×3 kernel, the 8P portion of this label indicates that the data have undergone 8×8 pooling, and the four at the end indicates that this label is at a distance of four from the input layer. The corresponding graphs, with assigned labels for each network, are shown in Figs. 13–20.

The next step is to put all these graphs in a parallel format sharing a single input and single output node. Figure 21 shows the graph in this step.

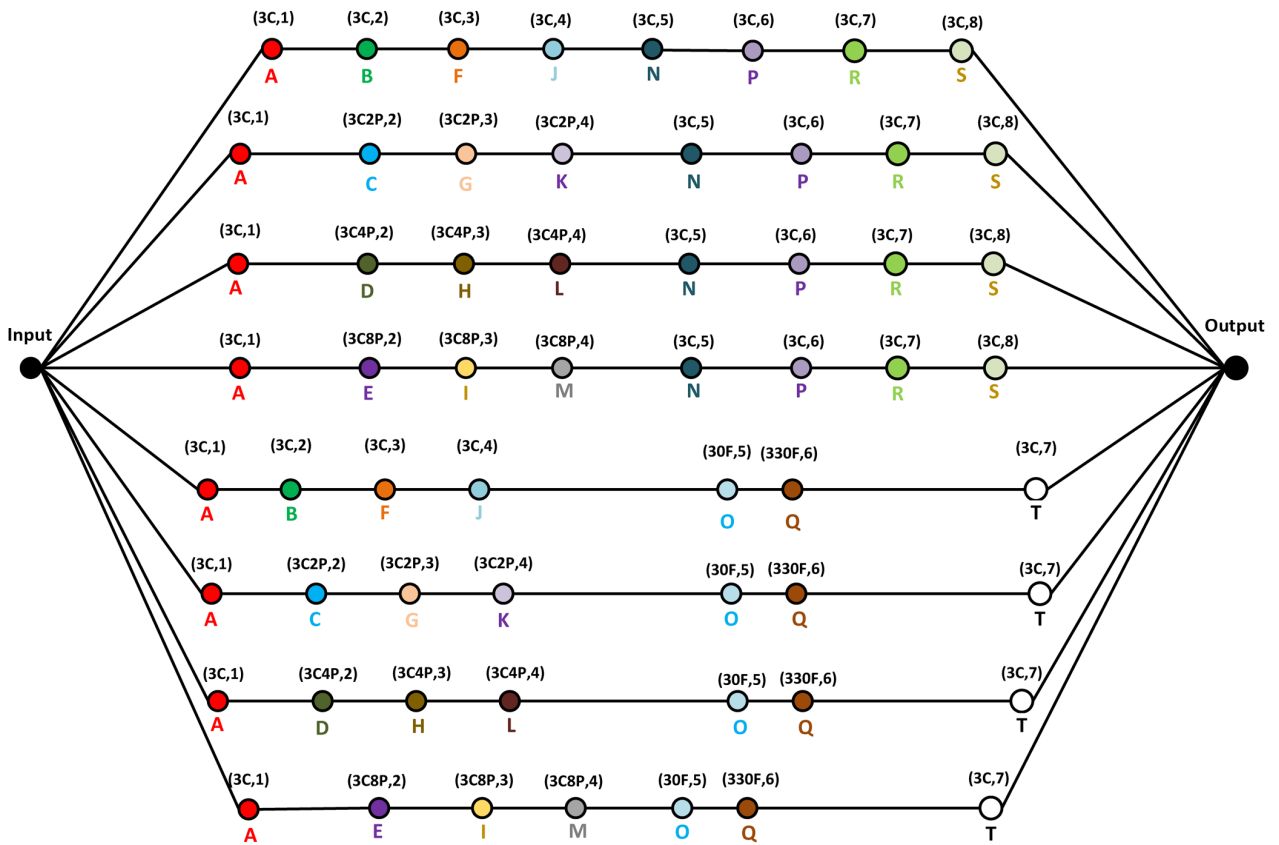


Fig. 21 Parallelized version of the graphs shown in Figs. 13–20 sharing a single input node and single output node.

In order to merge layers with the same structure and the same distance from the input node, nodes with the exact same properties are labeled with the same letters. For example, all the nodes with properties (3C, 1) are labeled with letter A, and all the nodes with the properties (3C2P, 4) are labeled K, and so on.

The next step is to apply graph contraction on the parallelized graph. In the graph contraction procedure, the nodes with the same label are merged to a single node while saving their connections to the previous/next nodes. For instance,

all the nodes with label A are merged into one node, but its connection to the input node and also nodes B, C, D, and E are preserved. The contracted version of the graph in Fig. 21 is shown in Fig. 22.

Afterward, the graph has to be converted back to the NN structure. In order to do this, the preserved structural properties of each node are used. For example, node C is a 3×3 convolutional layer that has experienced a pooling operation. Note that the pooling quality will be recalled from the original network.

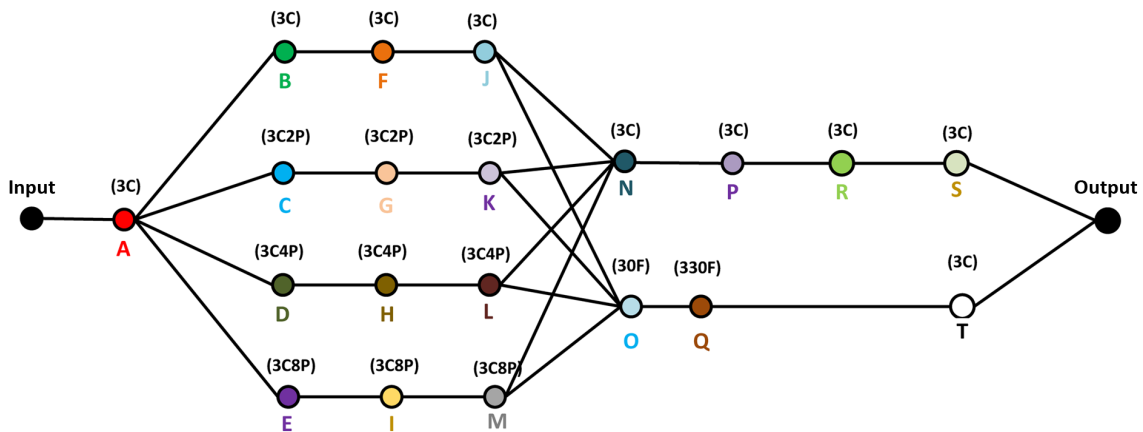


Fig. 22 Contracted version of the big graph shown in Fig. 21.

The concatenation layer is used in the NN in order to implement the nodes wherein several other nodes lead to one node. For example, in nodes N and O the outputs of nodes J, K, L, and M are concatenated with the pooling qualities taken from their original networks.

The graph is translated back to a DNN. The network corresponds to the graph shown in Fig. 22.

A.3 SPDNN: How It Works and Why It Is Effective?

One might ask why the SPDNN approach is effective and what the difference is between this approach and other mixing approaches. Here, the model designed by the SPDNN scheme is investigated in the forward and backpropagation steps. The key component is in the backpropagation step where the parameters in parallel layers influence each other. These two steps are described below:

Forward propagation: Consider the network designed by the SPDNN approach shown in Fig. 23. This exemplary network is made of five subnetworks. Just the general view of the network is shown in this figure and the layers' details are ignored since the main goal is to show the information flow within the whole network.

When the input samples are fed into the network, the data travel through the network along three different paths shown in Fig. 24.

At this stage, the parallel networks are blind to each other, i.e., the networks placed in parallel do not share any information with each other. As shown in Fig. 24, the data traveling in Sub-Net 1 and Sub-Net 2 are not influenced by each other since they do not share any path together, as in Sub-Net 3 and Sub-Net 4.

Backpropagation: While training the network, the loss function calculated based on the error value at the output of the NN is a mixed and merged function of the error value corresponding to every data path in the network. In the backpropagation step, the parameters inside the network update based on this mixed loss values, i.e., this value backpropagates throughout the whole network as shown in Fig. 25. Therefore, at this stage of training, each subnetwork is influenced by the error value from every data path shown

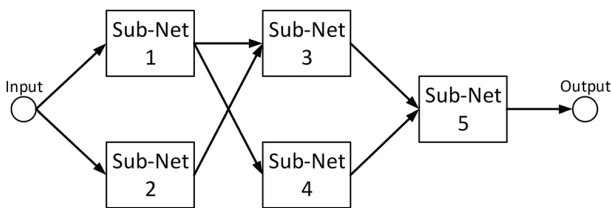


Fig. 23 A network designed using the SPDNN approach. It contains five subnetworks placed in parallel and semiparallel forms.

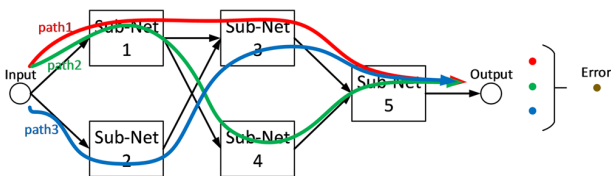


Fig. 24 Forward propagation inside the SPDNN. There are three different paths on which the information can flow inside the network.

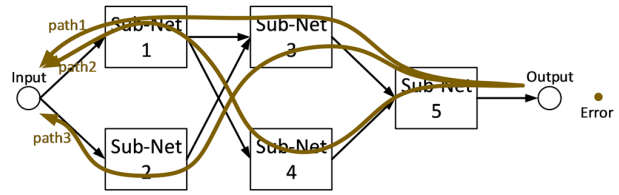


Fig. 25 Backpropagation for SPDNN. The mixed error is backpropagated throughout the network while updating parameters.

in Fig. 25. This illustrates the way each subnetwork is trained to reduce the error of its own path and also the error from the mixture of all paths.

The main difference between the SPDNN approach and other mixing approaches, such as the voting approach, lies in the backpropagation step where different subnets are influenced by the errors of each other and try to compensate for each other's shortcomings by reducing the final mixed error value. In the voting approach, different classifiers are trained independently of each other and they do not communicate to reduce their total error value.

A.3.1 SPDNN versus Inception

One of the approaches that has superficial similarities to SPDNN is the inception technique.⁶⁴ For clarity, and to aid the reader in understanding, the authors list four significant points of difference between SPDNN and inception with regard to mixing networks.

1. The main idea in SPDNN is to maintain the overall structure of the networks, but to mix them in a reasonable way. For example, if there is a big kernel such as 13×13 in one of the configurations, the SPDNN method always preserves the structure (13×13 kernel) inside the final network. This contrasts with inception,⁶⁴ which reduces larger kernels into smaller ones.
2. In the inception method, all the layers are merged into one final layer, which does not happen with the SPDNN approach.
3. The number of the layers in the SPDNN architecture is less than or equal to the number of layers in the original networks. In contrast, the inception idea aims to increase the number of layers in the network by (it breaks down each layer into several layers with smaller kernels).

The SPDNN idea is to design a new network from existing networks that perform well at some task or subtask while the idea in inception is to design a network from scratch.

A.4 Comparisons of Individual Networks

In this section, the behavior of each subnetwork is investigated and compared with the final network. Each of the eight networks proposed in Appendix A.2 is trained on the training data explained in Sec. 2.2. The training is performed in the Lasagne library on top of Theano in Python. Training is done on a standard desktop with an NVIDIA GTX 1080 GPU with 8 GB memory.

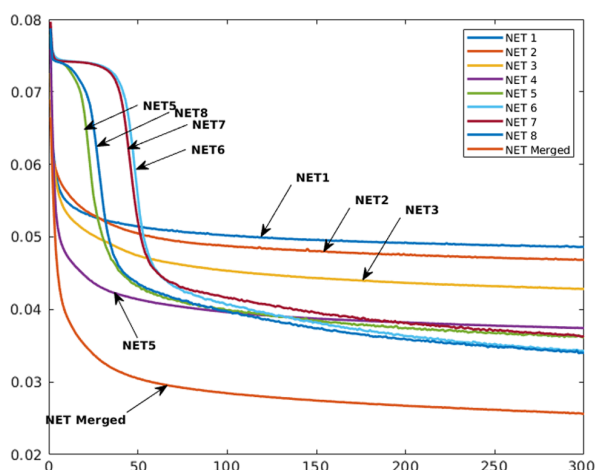


Fig. 26 Training loss for each subnetwork and also the merged network.

In the presented experiments, the MSE value between the output of the network and the target values has been used as the loss function, and the Nesterov momentum technique with learning rate 0.01 and momentum 0.9 has been used to train

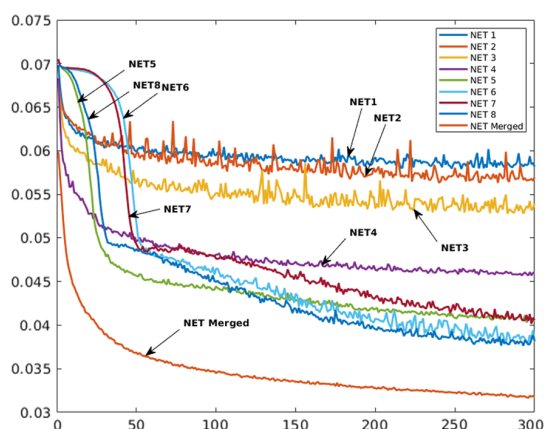


Fig. 27 Validation loss for each subnetwork and also the merged network.

the network. The training and validation losses for the first 300 epochs are shown Figs. 26 and 27, respectively.

The convergence of the network is significantly increased after merging the networks for both training and validation sets. The fluctuations in validation are less for the merged network, which demonstrates less variance in the loss value. This shows a more stable training process when the SPDNN is applied.

This also shows how much pooling can help the network to converge and also that networks with fully connected layers are providing better overall outputs, but they miss details in the depth maps since they observe the input sample as a single entity.

The evaluation of each network on the test set is presented in Table 10.

The merged network is giving superior results compared to each individual network for all measurements. This is while the final merged network is designed to have the same number of parameters as each individual subnetwork. This means that the same memory efficiency and also the processing speed for the merged network stay same as for the subnetworks.

Appendix B: SegNet

SegNet is a fully convolutional semantic image segmentation framework presented in Refs. 51 and 52. This model uses the convolutional layers of the VGG16 network as the encoder of the network and eliminates the fully connected layers, thus reducing the number of trainable parameters from 134 M to 14.7 M, which represents a reduction of 90% in the number of parameters to be trained. The encoder portion of SegNet consists of 13 convolutional layers with ReLU nonlinearity followed by max-pooling (2 × 2 window) and stride 2 in order to implement a nonoverlapping sliding window. This consecutive max-pooling and striding results in a network configuration that is highly robust to translation in the input image but has the drawback of losing spatial resolution of the data.

This loss of spatial resolution is not beneficial in segmentation tasks where it is necessary to preserve the boundaries of the input image in the segmented output. To overcome this problem, the following solution is given in Ref. 51. As most of the spatial resolution information is lost in

Table 10 Test loss for each subnetwork.

	Net #1	Net #2	Net #3	Net #4	Net #5	Net #6	Net #7	Net #8	Net merged
PSNR	12.2409	12.0153	12.3419	12.4382	12.1458	13.0247	12.3698	13.2233	15.0418
MSE	0.0640	0.0672	0.0639	0.0628	0.0676	0.0556	0.0639	0.0527	0.0378
RMSE	0.2486	0.2549	0.2471	0.2447	0.2535	0.2293	0.2467	0.2238	0.1854
SNR	0.8944	1.2153	1.3206	1.5260	2.5726	0.8139	1.5435	0.9104	8.822
MAE	0.2028	0.2081	0.2007	0.1938	0.2009	0.1864	0.2001	0.1798	0.1442
SSIM	0.9918	0.9914	0.9918	0.9922	0.9916	0.9927	0.9918	0.9932	0.9952
UQI	0.0455	0.0526	0.0856	0.1044	0.0954	0.1067	0.1154	0.1015	0.8401

the max-pooling operation, saving the information of the max-pooling indices and using this information in the decoder part of the network preserves the high-frequency information.

Note that for each layer in the encoder portion of the network, there is a corresponding decoder layer. The idea of SegNet is that wherever max-pooling is applied to the input data, the index of the feature with the maximum value is preserved. Later these indices will be employed to make a sparse feature space before the deconvolution step, applying the unpooling step in the decoder part. A batch normalization layer³⁵ is placed after each convolutional layer to avoid overfitting and to promote faster convergence. Decoder filter banks are not tied to corresponding encoder filters and are trained independently in the SegNet architecture.

Acknowledgments

The research work presented here was funded under the Strategic Partnership Program of Science Foundation Ireland (SFI) and co-funded by SFI and FotoNation Ltd. Project ID: 13/SPP/I2868 on “Next Generation Imaging for Smartphone and Embedded Platforms.” This work is also supported by an Irish Research Council Employment Based Programme Award. Project ID: EBPPG/2016/280.

References

- F. Tombari, S. Mattoccia, and L. D. Stefano, “Stereo for robots: quantitative evaluation of efficient and low-memory dense stereo algorithms,” in *11th Int. Conf. on Control Automation Robotics and Vision* (2010).
- S. Yang et al., “Extraction of topographic map elements with SAR stereoscopic measurement,” in *Int. Symp. on Image and Data Fusion* (2011).
- R. Muñoz-Salinas, E. Aguirre, and M. García-Silvente, “People detection and tracking using stereo vision and color,” *Image Vision Comput.* **25**(6), 995–1007 (2007).
- P. Haigron et al., “Depth-map-based scene analysis for active navigation in virtual angiography,” *IEEE Trans. Med. Imaging* **23**(11), 1380–1390 (2004).
- G. Yahav, G. J. Iddan, and D. Mandelbroum, “3D imaging camera for gaming application,” in *Digest of Technical Papers Int. Conf. on Consumer Electronics* (2007).
- M. Grosse et al., “3D shape measurement of macroscopic objects in digital off-axis holography using structured illumination,” *Opt. Lett.* **35**(8), 1233–1235 (2010).
- P. Kauff et al., “Depth map creation and image-based rendering for advanced 3DTV services providing interoperability and scalability,” *Image Commun.* **22**(2), 217–234 (2007).
- P. Merkle et al., “The effect of depth compression on multiview rendering quality,” in *3DTV Conf.: The True Vision—Capture, Transmission and Display of 3D Video* (2008).
- S. R. Malireddi et al., “HandSeg: a dataset for hand segmentation from depth images,” arXiv:171105944 (2017).
- L. Tian et al., “Robust 3D human detection in complex environments with depth camera,” *IEEE Trans. Multimedia* **1** (2018).
- J. Liu et al., “Skeleton-based human action recognition with global context-aware attention LSTM networks,” *IEEE Trans. Image Process.* **27**(4), 1586–1599 (2018).
- J. Liu et al., “Skeleton-based action recognition using spatio-temporal LSTM network with trust gates,” *IEEE Trans. Pattern Anal. Mach. Intell.* **1** (2017).
- M. Weber, M. Humenberger, and W. Kubinger, “A very fast census-based stereo matching implementation on a graphics processing unit,” in *IEEE 12th Int. Conf. on Computer Vision Workshops, ICCV Workshops* (2009).
- S. Bazrafkan and P. Corcoran, “Semi-parallel deep neural networks (SPDNN), convergence and generalization,” arXiv:171101963 (2017).
- S. Bazrafkan and P. M. Corcoran, “Pushing the AI envelope: merging deep networks to accelerate edge artificial intelligence in consumer electronics devices and systems,” *IEEE Consum. Electron. Mag.* **7**(2), 55–61 (2018).
- B. Freedman et al., “Depth mapping using projected patterns,” Google Patents, Patent No. US8493496B2 (2013).
- A. Govari et al., “Tissue depth estimation using gated ultrasound and force measurements,” Google Patents, Patent No. EP3189785A1 (2016).
- D. Nair, “3D Imaging with NI LabVIEW,” <http://www.ni.com/white-paper/14103/en/> (24 August 2016).
- C. Niclass et al., “A 100 m-range 10-frame/s 340 × 96-pixel time-of-flight depth sensor in 0.18- μ m CMOS,” in *Proc. of the ESSCIRC* (ESSCIRC) (2011).
- C. Niclass et al., “Design and characterization of a 256 × 64-pixel single-photon imager in CMOS for a MEMS-based laser scanning time-of-flight sensor,” *Opt. Express* **20**(11), 11863–11881 (2012).
- D. Scharstein and R. Szeliski, “High-accuracy stereo depth maps using structured light,” in *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition* (2003).
- R. K. Gupta and S.-Y. Cho, “A correlation-based approach for real-time stereo matching,” *Lect. Notes Comput. Sci.* **6454**, 129–138 (2010).
- C. Zhang et al., “MeshStereo: a global stereo model with mesh alignment regularization for view interpolation,” in *IEEE Int. Conf. on Computer Vision (ICCV)* (2015).
- D. Scharstein and R. Szeliski, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *Int. J. Comput. Vision* **47**(1–3), 7–42 (2002).
- K. R. Kim and C. S. Kim, “Adaptive smoothness constraints for efficient stereo matching using texture and edge information,” in *IEEE Int. Conf. on Image Processing (ICIP)* (2016).
- X. Huang, Y. Zhang, and Z. Yue, “Image-guided non-local dense matching with three-steps optimization,” *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* **III-3**, 67–74 (2016).
- A. Li et al., “Coordinating multiple disparity proposals for stereo computation,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada (2016).
- M. Shahbazi et al., “Revisiting intrinsic curves for efficient dense stereo matching,” *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* **III-3**, 123–130 (2016).
- J. Žbontar and Y. LeCun, “Stereo matching by training a convolutional neural network to compare image patches,” *J. Mach. Learn. Res.* **17**(1), 2287–2318 (2016).
- J. T. Barron and B. Poole, “The fast bilateral solver,” arXiv:151103296 (2016).
- E. T. Psota et al., “MAP disparity estimation using hidden Markov trees,” in *IEEE Int. Conf. on Computer Vision (ICCV)* (2015).
- J. Kowalczyk, E. T. Psota, and L. C. Perez, “Real-time stereo matching on CUDA using an iterative refinement method for adaptive support-weight correspondences,” *IEEE Trans. Circuits Syst. Video Technol.* **23**(1), 94–104 (2013).
- B. Yegnanarayana, *Artificial Neural Networks*, PHI Learning Pvt. Ltd., New Delhi (2009).
- N. Srivastava et al., “Dropout: a simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014).
- S. Ioffe and C. Szegedy, “Batch normalization: accelerating deep network training by reducing internal covariate shift,” arXiv:150203167 (2015).
- F. Mahmood, R. Chen, and N. J. Durr, “Unsupervised reverse domain adaptation for synthetic medical images via adversarial training,” *IEEE Trans. Medical Imaging* **1** (2018).
- F. Mahmood and N. J. Durr, “Deep learning-based depth estimation from a synthetic endoscopy image training set,” *Proc. SPIE* **10574**, 1057421 (2018).
- F. Mahmood and N. J. Durr, “Deep learning and conditional random fields-based depth estimation and topographical reconstruction from conventional endoscopy,” arXiv:171011216 (2017).
- F. Liu et al., “Learning depth from single monocular images using deep convolutional neural fields,” *IEEE Trans. Pattern Anal. Mach. Intell.* **38**(10), 2024–2039 (2016).
- A. Saxena, J. Schulte, and A. Y. Ng, “Depth estimation using monocular and stereo cues,” in *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence*, Hyderabad, India (2007).
- D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in *Proc. of the 27th Int. Conf. on Neural Information Processing Systems*, Montreal, Canada, Vol. 2 (2014).
- R. Garg et al., “Unsupervised CNN for single view depth estimation: geometry to the rescue,” *Lect. Notes Comput. Sci.* **9912**, 740–756 (2016).
- C. Godard, O. M. Aodha, and G. J. Brostow, “Unsupervised monocular depth estimation with left-right consistency,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2017).
- D. Xu et al., “Monocular depth estimation using multi-scale continuous CRFs as sequential deep networks,” *IEEE Trans. Pattern Anal. Mach. Intell.* **1** (2018).
- D. Xu et al., “Multi-scale continuous CRFs as sequential deep networks for monocular depth estimation,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2017).
- N. Silberman et al., “Indoor segmentation and support inference from RGBD images,” *Lect. Notes Comput. Sci.* **7576**, 746–760 (2012).

47. M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2015).
48. A. Saxena, S. H. Chung, and A. Y. Ng, "3-D depth reconstruction from a single still image," *Int. J. Comput. Vision* **76**(1), 53–69 (2008).
49. A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *Proc. of the 18th Int. Conf. on Neural Information Processing Systems*, Vancouver, British Columbia (2005).
50. H. Javidnia and P. Corcoran, "A depth map post-processing approach based on adaptive random walk with restart," *IEEE Access* **4**, 5509–5519 (2016).
51. V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: a deep convolutional encoder-decoder architecture for image segmentation," arXiv:1511.00561 (2015).
52. A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segNet: model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," arXiv:1511.02680 (2015).
53. A. Kendall, V. Badrinarayanan, and R. Cipolla, "Caffe implementation of SegNet," <https://github.com/alexgkendall/caffe-segnet> (18 April 2018).
54. G. J. Brostow et al., "Segmentation and recognition using structure from motion point clouds," *Lect. Notes Comput. Sci.* **5302**, 44–57 (2008).
55. S. Lee et al., "Robust stereo matching using adaptive random walk with restart algorithm," *Image Vision Comput.* **37**, 1–11 (2015).
56. I. Sutskever et al., "On the importance of initialization and momentum in deep learning," in *Int. Conf. Machine Learning*, Vol. 28, pp. 1139–1147 (2013).
57. W. Zhou et al., "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.* **13**(4), 600–612 (2004).
58. W. Zhou and A. C. Bovik, "A universal image quality index," *IEEE Signal Process. Lett.* **9**(3), 81–84 (2002).
59. K. Pearson, "Note on regression and inheritance in the case of two parents," *Proc. R. Soc. Lond.* **58**, 240–242 (1895).
60. J. Kopf et al., "Joint bilateral upsampling," *ACM Trans. Graph* **26**(3), 96 (2007).
61. A. Saxena, M. Sun, and A. Y. Ng, "Make3D: learning 3D scene structure from a single still image," *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(5), 824–840 (2009).
62. T. Zhou et al., "Unsupervised learning of depth and ego-motion from video," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2017).
63. Y. Kuznetsov, J. Stückler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2017).
64. C. Szegedy et al., "Going deeper with convolutions," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition* (2015).

Shabab Bazrafkan received his BSc degree in electrical engineering from Urmia University, Urmia, Iran, in 2011 and his MSc degree from Shiraz University of Technology (SuTECH) in telecommunication engineering, image processing branch in 2013. Currently he is a PhD student at the National University of Ireland, Galway (NUIG) and also works with Fotonation Ltd. His field of working is deep neural networks and neural network design.

Hossein Javidnia received his master's degree in information technology engineering from the University of Guilan, Iran, in 2014. He received his PhD in electrical engineering from the National University of Ireland, Galway, in 2018. His current research interests include image processing, machine vision, and automotive navigation.

Joseph Lemley received his BS degree in computer science from Central Washington University in 2006. After working in industry and cofounding a start-up, he went back to Central Washington University in 2014 and received his master's degree in computational science in 2016. Currently he is a PhD student at the National University of Ireland, Galway (NUIG) funded by Fotonation Ltd. under the IRCSET "Employment PhD" program. His field of work is machine learning using deep neural networks for tasks related to computer vision.

Peter Corcoran is an IEEE fellow; 500+ technical publications and patents; 80+ peer reviewed papers and articles; 100+ international conference papers; coinventor on 300+ granted US patents, university professor for 28 years; member IEEE Consumer Electronics Society 20+ years; editor-in-chief and founding editor of IEEE Consumer Electronics Magazine; former vice-dean of research and graduate studies (7 year tenure) in the College of Engineering and Informatics at NUI Galway; cofounder of several start-up companies including FotoNation (www.fotonation.com); and industry consultant and expert witness.

Appendix C

**Deep Learning for Facial Expression
Recognition: A step closer to a
SmartPhone that Knows your Moods**

Deep Learning for Facial Expression Recognition: *A step closer to a SmartPhone that Knows your Moods.*

Shabab Bazrafkan¹, Tudor Nedelcu¹, Pawel Filipczuk², Peter Corcoran¹ *Member, IEEE*

1: Center for Cognitive, Connected & Computational Imaging, College of Engineering & Informatics,
NUI Galway, Galway, Ireland

2: Fotonation LTD, Galway, Ireland

E-mails: {s.bazrafkan1, t.nedelcu}@nuigalway.ie, pfilipczuk@fotonation.com,
peter.corcoran@nuigalway.ie

Abstract—By growing the capacity and processing power of the handheld devices nowadays, a wide range of capabilities can be implemented in these devices to make them more intelligent and user friendly. Determining the mood of the user can be used in order to provide suitable reactions from the device in different conditions. One of the most studied ways of mood detection is by using facial expressions, which is still one of the challenging fields in pattern recognition and machine learning science.

Deep Neural Networks (DNN) have been widely used in order to overcome the difficulties in facial expression classification. In this paper it is shown that the classification accuracy is significantly lower when the network is trained with one database and tested with a different database. A solution for obtaining a general and robust network is given as well.

I. INTRODUCTION

Today's handheld devices are growing in their capacity to interact with end-users. They have access to an ever-growing range of network based services and their sensing capabilities of the location and local environment continue to grow in scope. One remaining challenge for today's devices is to sense and determine the emotional state of the user. This introduces new challenges [1], [2] and requires a range of sophisticated edge technologies that can capture and analyze information from the user on the device. One example is the real-time analysis of speech patterns for detecting emotion [3], [4]. More recently researchers in this field have turned to deep learning techniques [5]. Facial expression analysis is also well known in the literature [6]–[11].

But it is computationally complex and it is challenging to achieve high recognition rates using conventional feature extraction and classification schemes. In this paper we follow the trend from the speech recognition field and explore facial emotion recognition using deep learning techniques. The goal is to demonstrate the potential for high performance solution that can run on relative lightweight convolutional neural networks that can be efficiently implemented in hardware or on a GPU. Such a solution could realistically enable a new generation of smartphones that can understand the moods of their owners.

This research is funded under the *SFI Strategic Partnership Program* by Science Foundation Ireland (SFI) and FotoNation Ltd. Project ID: 13/SPP/12868 on *Next Generation Imaging for Smartphone and Embedded Platforms*.

This work is supported by the Enterprise Partnership Scheme program of the Irish Research Council

A. Facial Expression Classification

This In recent years the facial expressions classification has attracted a lot of attention because of its various potential applications including psychology, medicine, security [12], man-machine interaction and surveillance [13]. There are two main approaches to investigate the facial expression in a systematic way: Action Unit (AU) based and appearance based methods.

AU model introduces the Facial Action Coding System (FACS) which has been developed by Carl-Herman Hjortsjö in 1969 [14]. This technique described the facial expression as a composition of Action Units which are describing the facial muscle motions. This method takes advantage of the strong support of the psychology and physiology sciences since it uses the facial muscle movements for modeling different expressions [13]. The AU based methods suffer from the difficulties such as dependencies on invisible muscle motions [13] which makes it extremely difficult to model the FACS system using machines.

In contrast the appearance based methods are using feature extraction, feature selection and classification methods [15] in order to determine the expression in the face. In this approach different kinds of features have been used so far including, Local Binary Pattern [16], Scale Invariant Feature Transform (SIFT) [17] and Histogram of Oriented Gradient [18]. There are several main difficulties in facial expression detection, like, changes in the appearance and the shape of the face in unexpected ways (because of the non-rigidity of the face) [19], imaging conditions, and inter-person differences in facial expressions. Because of these basic problems there are no golden methods which can be called as a standard for automatic facial expression classification.

B. Deep Learning

In recent years by emerging powerful parallel processing hardware, Deep Neural Networks (DNN) become a hot topic in pattern recognition and machine learning science. Deep Learning (DL) scheme is based on the consecutive layers of signal processing units in order to mix and re-orient the input data to their most representative order correspond to a specific application.

Facial expression classification has taken advantage of DNN classifiers in recent years. In [13] an AU inspired Deep Network has been proposed which uses the DNN to extract the most representative features. In [20] a DNN with five layers

and 65K neurons has been designed to classify the expression into five categories (Neutral, happy, sad, angry and surprised). In [15], a Boosted Deep Belief Network (BDBN) has been proposed and implemented using joint fine tune process in BDBN framework to classify the facial expression.

In all DNN based investigations on facial expression, the proposed network is trained and tuned for a specific database and the test data is drawn from the same database as well. Since the network is biased for a specific data type, the result on the test data of that dataset will give surprisingly low error rate. But if the designed network for database A would be tested on database B the results will be shockingly bad. Therefore, these classifiers would fail to work in wild environments.

The main goal of this paper is to investigate the amount of error caused by network trained with one database and tested with other and also design and implement a network which can overcome the problem with mixing databases for training stage.

In the next section an introduction to Deep Neural Networks (DNN) is presented, and also Databases and database expansion is presented. In the Third section the networks designed for single and multi database purposes are given and results and discussion is given in section 4.

II. METHODOLOGY

The goal of this research is to investigate the amount of error that occurs from training a DNN network for a database and test it on other database. This inter-database investigation can give a general perspective on design and train networks for wild applications and costumer device implementations.

A. Deep Neural Networks (DNN)/Deep Learning (DL)

Deep Neural Networks training also known as Deep Learning is one of the most advanced machine learning techniques trending in recent years due to appearance of extremely powerful parallel processing hardware and Graphical Processing Units (GPU). Several consecutive signal processing units are set in serial/parallel architecture mixing and re-orienting the input data in order to result in most representative output considering a specific problem. The most popular image/sound processing structure of DNN is constructed by three main processing layers: Convolutional Layer, Pooling Layer and Fully Connected Layer. DNN units are described below:

Convolutional Layer: This layer convolves the (in general 3D) image “I” with (in general 4D) kernel “W” and adds a (in general 3D) bias term “b” to it. The output is given by:

$$P = I * W + b, \quad (1)$$

where * operator is nD convolution in general. In the training process the kernel and bias parameters are selected in a way to optimize the error function of the network output.

Pooling Layer: The pooling layers applies a non-linear transform on the input image which reduce the neuron numbers after the operation. It’s common to put a pooling layer between two consecutive convolutional layers. This operation also reduces the unit size which will lead to less computational load and also prevents the over-fitting problem.

Fully Connected Layer: Fully connected layers are exactly same as the classical Neural Network (NN) layers where all the neurons in a layer are connected to all the neurons in their subsequent layer. The neurons are triggered by the summation of their input multiplied by their weights passed from their activation functions.

B. Databases

Three databases has been used in the research. Radboud Faces Database (RaFD) [21], Cohn-Kanade AU-Coded Facial Expression Database Version 2 (CK+) [22] and The Japanese Female Facial Expression (JAFFE) Database [23].

RaFD: The Radboud Faces Database is a set of 67 persons with different gender and different races (Caucasian and Moroccan Dutch), both children and adults. This database displays 8 different emotions which in the presented work seven of them are used.

CK+: Cohn-Kanade version 2 (known as CK+) database is a facial expression database including both posed and non-posed expressions wherein the subject changes emotion in several sequences from neutral to one of seven different expressions. In the presented work just the non-posed data has been used.

JAFFE: The Japanese Female Facial Expression Database is made of 213 images of 7 expressions contains faces of 10 Japanese female models. Since the number of images in this database is not sufficient to train a DNN, this database is eliminated from our inter-database investigations and is used just for multi database network training.

C. Database Expansion

Since the number of image sin each database is not enough in order to train a DNN a database expansion scheme has been used to overcome the problem which includes flipping images and rotating them by [-3,-2,-1,1,3,3] degrees and put them back in the dataset. Using this approach, we ended up with large number of images for each dataset shown in table 1.

Table 1: Number of images in each database and each dataset

Database\Dataset	Train	Validation	Test
RaFD	13160	312	146
CK+	14392	304	187
JAFFE	1960	42	22

As it has been shown in table 1, we can see that there are not enough images in JAFFE database to train a DNN. This database is used to train the multi-database network. Just note that the database expansion applied to training samples and Validation and Test samples remain unchanged.

III. DEEP NEURAL NETWORKS FOR EXPRESSION CLASSIFICATION

A. Single database networks

For each of databases RaFD and CK+ a DNN has been designed and trained and the results for testing on each database is calculated as well. The networks designed for each database are given in the following list.

DNN on RaFD: The architecture for network trained on RaFD database is given in table 2.

Table 2: Network configuration for RaFD database

layer	Kernel/Units	Size/Dropout Probability
Convolutional	16	3x3
Maxpool	N/A	2x2
Convolutional	8	3x3
Maxpool	N/A	2x2
Convolutional	8	3x3
Maxpool	N/A	2x2
Fully Connected	15	Dropout p=0.8
Fully Connected	7	Dropout p=0.5

DNN on CK+: The architecture for network trained on CK+ database is given in table 3.

Table 3: Network Configuration for CK+ database.

Layer	Kernel/Units	Size/Dropout Probability
Convolutional	8	3x3
Maxpool	N/A	2x2
Convolutional	8	3x3
Maxpool	N/A	2x2
Convolutional	8	3x3
Maxpool	N/A	2x2
Fully Connected	7	Dropout p=0.5

B. Multi-Database Network

A network has been designed and trained for a mixture of all three databases. The architecture of the network is given in table 4.

Table4: Network Configuration for mixed Dataset

Layer	Kernel/Units	Size/Dropout Probability
Convolutional	16	3x3
Maxpool	N/A	2x2
Convolutional	13	3x3
Maxpool	N/A	2x2
Convolutional	10	3x3
Maxpool	N/A	2x2
Fully Connected	7	Dropout p=0.5

IV. RESULTS AND DISCUSSION

Three networks explained in the previous section have been implemented using Lasagne library (a Theano based DNN library in python). The mean categorical cross-entropy loss

function has been used with Nesterov momentum optimization. First and second networks shown in tables 2 and 3 are trained by RaFD and CK+ datasets respectively and the third network shown in table 4 is trained using a mixture of three databases RaFD, CK+ and JAFFE. The main goal of this work is to present the cross-database error which is accomplished by calculating the amount of error for each network using all databases. The classification error is presented in table 5.

Table5: Error given from each network for each dataset

Network\error	Error for RaFD	Error for CK+	Error for JAFFE
Network 1	6.84%	59.59%	72.73%
Network 2	21.23%	19.59%	50%
Network 3	4.1%	16.04%	13.36%

The first two rows of the table 5 are associated with the classification test error values for networks trained just with one database; RaFD for network 1 and CK+ for network 2. Since these networks are trained with just one database, they are tuned for that specific database and the test error for that specific network is less than the errors for other databases. Network 3 is trained with a mixture of three databases given in section II.B.

The most important result is that in network 3 the test error for each database is even lower than the value of the error from the network which is trained by that specific database.

While implementing a solution in consumer devices, it is crucial to provide algorithms which are robust to environment and condition changes. In this work it has been shown that in order to obtain a more general and robust DNN, one of the solutions is to mix as much data as possible. In fact adding a wide range of samples drawn from different conditions and properties to DNN training sets, will lead to a more reliable network for wild use cases which is one of the most important considerations in consumer electronic devices.

REFERENCE

- [1] V. Pejovic and M. Musolesi, "Anticipatory mobile computing: A survey of the state of the art and research challenges," *ACM Comput. Surv.*, vol. 47, no. 3, pp. 1–29, 2015.
- [2] R. Rana, M. Hume, J. Reilly, R. Jurdak, and J. Soar, "Opportunistic and Context-Aware Affect Sensing on Smartphones," *IEEE Pervasive Comput.*, vol. 15, no. 2, pp. 60–69, 2016.
- [3] M. El Ayadi, M. S. Kamel, and F. Karray, "Survey on speech emotion recognition: Features, classification schemes, and databases," *Pattern Recognit.*, vol. 44, no. 3, pp. 572–587, 2011.
- [4] C. N. Anagnostopoulos, T. Iliou, and I. Giannoukos, "Features and classifiers for emotion recognition from speech: a survey from 2000 to 2011," *Artif. Intell. Rev.*, vol. 43, no. 2, pp. 155–177, 2012.
- [5] R. Rana, R. Jurdak, X. Li, and J. Soar, "Emotion Classification from Noisy Speech-A Deep Learning Approach," *arXiv Prepr. arXiv*, 2016.
- [6] J. Sung and D. Kim, "Pose-Robust Facial Expression Recognition Using View-Based 2D + 3D AAM," *Syst. Man Cybern. Part A Syst. Humans, IEEE Trans.*, vol. 38, no. 4, pp. 852–866, 2008.
- [7] I. Bacivarov and P. Corcoran, "Facial expression modeling using component AAM models—Gaming applications," in *Games Innovations Conference, International IEEE Consumer Electronics Society's. (ICE-GIC 2009)*, 2009, pp. 1–16.

- [8] G. Mancini, S. Agnoli, B. Baldaro, P. E. R. Bitti, and P. Surcinelli, "Facial expressions of emotions: recognition accuracy and affective reactions during late childhood," *J. Psychol.*, vol. 147, no. 6, pp. 599–617, 2013.
- [9] S. L. Happy and A. Routray, "Automatic facial expression recognition using features of salient facial patches," *IEEE Trans. Affect. Comput.*, vol. 6, no. 1, pp. 1–12, 2015.
- [10] L. Ding and A. M. Martinez, "Features versus context: An approach for precise and detailed detection and delineation of faces and facial features," *Pattern Anal. Mach. Intell. IEEE Trans.*, vol. 32, no. 11, pp. 2022–38, Nov. 2010.
- [11] I. Bacivarov, "Advances in the Modelling of Facial Sub-Regions and Facial Expressions using Active Appearance Techniques," National University of Ireland Galway, 2009.
- [12] O. Rudovic, M. Pantic, and I. Patras, "Coupled Gaussian processes for pose-invariant facial expression recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1357–1369, 2013.
- [13] M. Liu, S. Li, S. Shan, and X. Chen, "AU-inspired Deep Networks for Facial Expression Feature Learning," *Neurocomputing*, vol. 159, no. 1, pp. 126–136, 2015.
- [14] C.-H. Hjortsjö, *Man's face and mimic language*. Studen litteratur, 1969.
- [15] P. Liu, S. Han, Z. Meng, and Y. Tong, "Facial Expression Recognition via a Boosted Deep Belief Network," *2014 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1805–1812, 2014.
- [16] C. Shan, S. Gong, and P. W. McOwan, "Facial expression recognition based on local binary patterns: A comprehensive study," *Image Vis. Comput.*, vol. 27, no. 6, pp. 803–816, 2009.
- [17] U. Tariq, K.-H. Lin, Z. Li, X. Zhou, Z. Wang, V. Le, T. S. Huang, X. Lv, and T. X. Han, "Emotion recognition from an ensemble of features," in *Automatic Face & Gesture Recognition and Workshops (FG 2011)*, *2011 IEEE International Conference on*, 2011, pp. 872–877.
- [18] Y. Hu, Z. Zeng, L. Yin, X. Wei, X. Zhou, and T. S. Huang, "Multi-view facial expression recognition," in *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, 2008, pp. 1–6.
- [19] Y. Wu, Z. Wang, and Q. Ji, "Facial feature tracking under varying facial expressions and face poses based on restricted boltzmann machines," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, pp. 3452–3459, 2013.
- [20] I. Song, H. J. Kim, and P. B. Jeon, "Deep learning for real-time robust facial expression recognition on a smartphone," *Dig. Tech. Pap. - IEEE Int. Conf. Consum. Electron.*, pp. 564–567, 2014.
- [21] O. Langner, R. Dotsch, G. Bijlstra, D. H. J. Wigboldus, S. T. Hawk, and A. van Knippenberg, "Presentation and validation of the Radboud Faces Database," *Cogn. Emot.*, vol. 24, no. 8, pp. 1377–1388, 2010.
- [22] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, 2010, pp. 94–101.
- [23] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, 1998, pp. 200–205.

Appendix D

An End to End Deep Neural Network for Iris Segmentation in Unconstraint Scenarios



An end to end Deep Neural Network for iris segmentation in unconstrained scenarios

Shabab Bazrafkan^{a,*}, Shejin Thavalengal^b, Peter Corcoran^a

^a Department of Electronic Engineering, College of Engineering, National University of Ireland Galway, University Road, Galway, Ireland

^b Xperi Galway, Cliona Building One, Parkmore East Business Park, Ballybrit, Galway, Ireland

ARTICLE INFO

Article history:

Received 8 December 2017
Received in revised form 8 May 2018
Accepted 21 June 2018
Available online 30 June 2018

Keywords:

Deep Neural Networks
Data augmentation
Iris segmentation

ABSTRACT

With the increasing imaging and processing capabilities of today's mobile devices, user authentication using iris biometrics has become feasible. However, as the acquisition conditions become more unconstrained and as image quality is typically lower than dedicated iris acquisition systems, the accurate segmentation of iris regions is crucial for these devices. In this work, an end to end Fully Convolutional Deep Neural Network (FCDNN) design is proposed to perform the iris segmentation task for lower-quality iris images. The network design process is explained in detail, and the resulting network is trained and tuned using several large public iris datasets. A set of methods to generate and augment suitable lower quality iris images from the high-quality public databases are provided. The network is trained on Near InfraRed (NIR) images initially and later tuned on additional datasets derived from visible images. Comprehensive inter-database comparisons are provided together with results from a selection of experiments detailing the effects of different tunings of the network. Finally, the proposed model is compared with SegNet-basic, and a near-optimal tuning of the network is compared to a selection of other state-of-art iris segmentation algorithms. The results show very promising performance from the optimized Deep Neural Networks design when compared with state-of-art techniques applied to the same lower quality datasets.

© 2018 Elsevier Ltd. All rights reserved.

1. Introduction

Biometric technology has become increasingly integrated into our daily life, from unlocking the smartphone to cash withdrawals from ATMs to shopping in the local supermarket (Pando, 2017). Various biometric modalities such as face, iris, retina, voice, fingerprints, palm prints, palm geometry are being used in a multitude of applications including law enforcement, border crossing and consumer applications (Corcoran & Costache, 2016; Thavalengal & Corcoran, 2016). The iris of the human eye – the annular region between the pupil and sclera – is of particular interest as iris is a biometric modality with high distinctiveness, permanence, and performance (Prabhakar, Pankanti, & Jain, 2003).

The historical evolution of Iris recognition systems can be broadly summarized by a number of key stages, each presenting a new set of unique challenges over earlier implementations of the technology:

- (i) The original proposal for the use of the iris as a biometric was made by the ophthalmologist Burch in 1936 (Irsch, Guyton & Johns, 2009) and the underlying technology to automate iris

recognition for practical deployment was proposed and subsequently developed by Daugman (1994) during the 1990s. Such systems acquired the iris pattern using a dedicated imaging system that constrained the target eye and employed near-infrared (NIR) imaging.

- (ii) Systems supporting the acquisition of iris patterns from mobile persons, in unconstrained acquisition conditions, were developed during the 2000s, with the Iris On The Move system from Sarnoff being one of the better known of these (Matey et al., 2006). This system was designed for deployment in public spaces such as airports and requires people to walk along a specified path where multiple successive iris images are acquired by a multi-camera systems under controlled lighting conditions.
- (iii) Most recently iris recognition has been developed and deployed on handheld devices including smartphones (Thavalengal & Corcoran, 2016). Such image acquisition is unsupervised and to a large extent unconstrained. This introduces new artifacts that are not found in earlier acquisition environments including unwanted reflections, occlusions, non-frontal iris images, low contrast and partially blurred images. Isolating iris regions accurately in such an acquisition environment has proved to be more challenging

* Corresponding author.

E-mail address: s.bazrafkan1@nuigalway.ie (S. Bazrafkan).

and requires improvements to the authentication workflow as will be discussed shortly.

The majority of existing iris recognition systems follow the authentication workflow as (i) image acquisition: an eye image is acquired using a camera, (ii) iris segmentation: eye/iris region is located in this image followed by isolating the region representing iris. (iii) Feature extraction: relevant features which represent the uniqueness of the iris pattern is extracted from the iris region and (iv) similarity of the two iris representation is evaluated by pattern matching techniques.

The work presented in this paper focuses on successful segmentation of non-ideal iris images, an essential element of the authentication workflow if an unacceptably high level of failed authentications is to be avoided.

1.1. Significance of iris segmentation

Iris segmentation involves the detection and isolation the iris region from an eye image. The subsequent feature extraction and pattern matching stages of any authentication workflow rely on the accurate segmentation of the iris and failed segmentations represent the single largest source of error in the iris authentication workflow (Bigun, Alonso-Fernandez, Hofbauer, & Uhl, 2016; Erbilek, Da Costa-Abreu, & Fairhurst, 2012; Proença & Alexandre, 2010). For an accurate segmentation, the exact iris boundaries at pupil and sclera have to be obtained, the occluding eyelids have to be detected, and reflections have to be removed, or flagged. Errors at the segmentation stage are propagated to subsequent processing stages (Bigun et al., 2016; Hofbauer, Alonso-Fernandez, Wild, Bigun, & Uhl, 2014). Detailed analysis of the impact of iris segmentation is studied in Bigun et al. (2016), Erbilek et al. (2012) and Proença and Alexandre (2010).

Numerous factors can introduce challenges in accurate iris segmentation (Jillela & Ross, 2013) even on high-resolution iris systems. Examples include (i) occlusions caused by the anatomical features of the eye; (ii) illumination conditions; (iii) user cooperation; (iv) environmental factors; (v) noise & manufacturing variations in image sensor technology; (vi) nature of the interacting population. These factors apply to all iris acquisition systems.

For mobile devices, in addition to these generic factors, there are additional concerns. Various image quality factors can also affect iris segmentation (Alonso-Fernandez & Bigun, 2013) and these become a limiting factor in consumer devices such as smartphones due to the challenging nature of acquiring suitable high-quality images in a user-friendly smartphone use-case (Thavalengal, Bi-gioi, & Corcoran, 2015a, b). Hence, an iris segmentation technique which can accurately isolate the iris region in such low-quality consumer images is important for the wider adoption constraint-free consumer iris recognition system.

This work proposes to significantly improve the quality of iris segmentation on lower quality images by introducing an end-to-end deep neural network model accompanied by an augmentation technique. These improvements should enable improved iris authentication systems for today's mobile devices, encouraging a broader adoption of iris recognition in day-to-day use cases.

1.2. Related literature & foundation methods

1.2.1. Iris segmentation

A detailed review of iris segmentation literature can be found in Bowyer, Hollingsworth, and Flynn (2008, 2013). Early work on iris segmentation approximated the pupillary and limbic boundaries as circles (Bowyer et al., 2008). An appropriate circular fitting method is incorporated for modeling these boundaries. Daugman's original work uses an integrodifferential operator for iris segmentation (Daugman, 2004). This integrodifferential operator acts as

a circular edge detector which searches over the image domain for the best circle fit. Applying this operator twice, one can obtain the two circular boundaries of iris. After this step, the occluding eyelashes are detected with the help of curvilinear edge detection. There have been several similar techniques for iris segmentation such as the algorithm proposed by Wildes et al. (1996), Kong and Zhang (2001), Tisse et al. (1992) and Ma, Wang, and Tan (2002). (All of these use circular Hough transform for finding the circles). Another segmentation technique proposed by He, Tan, Sun, and Qiu (2009) uses an Adaboost-cascade iris detector and an elastic model named 'pulling and pushing method'.

Further studies revealed that iris and pupil boundaries are not always circular, and modeling this accurately, improves the iris recognition performance (Daugman, 2007). Daugman's follow up work (Daugman, 2007) incorporates active contours or snakes to model the iris accurately. A similar approach proposed by Shah and Ross (2009) uses geodesic active contours for accurate iris segmentation. Such techniques were shown to have high segmentation accuracy in high quality images captured using dedicated iris cameras in the NIR region of electromagnetic spectrum. Proença and Alexandre noted the poor performance of iris segmentation techniques developed for high quality images when applied to non-ideal images (Proença & Alexandre, 2006). A recent literature survey on non-ideal iris image segmentation can be found in Jan (2017). Among the literature, it is worth to be noted the efforts of mobile iris challenge evaluation (MICHE) to provide a forum for comparative research on the contributions to the mobile iris recognition field (De Marsico, Nappi, Riccio, & Wechsler, 2015; Marsico, Nappi, & Proença, 2017). Techniques based on various adaptive filtering and thresholding approaches are shown to be performing well in these non-ideal scenarios (Gangwar, Joshi, Singh, Alonso-Fernandez, & Bigun, 2016; Haindl & Krupička, 2015).

1.2.2. Applications of CNNs in iris recognition

In the last decade, deep learning techniques have become the focus of intense research and the most successful approach in artificial intelligence and machine vision science. Deep learning based techniques are noted to provide state of the art results in various applications such as object detection (Szegedy, Toshev, & Erhan, 2013), face recognition (Schroff, Kalenichenko, & Philbin, 2015), driver monitoring systems (Lemley, Bazrafkan, & Corcoran, 2017a, b, c), etc. In such deep learning based approaches, the input signal (image) is processed by consecutive signal processing units. These units re-orient the input data to the most representative shape considering the target samples. The signal processing units are known as layers which could be convolutional or fully connected. The fully convolutional models, such as the one presented in this work, are using only convolutional layers. These layers apply filters (known as kernels) to their input while the filter parameters are learned in the training step. In order to get better convergence, several techniques including drop-out (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) and batch normalization (Ioffe & Szegedy, 2015) are presented in the literature. A detailed introduction to CNNs (Convolutional Neural Networks) and its applications can be found in (Lemley et al., 2017a, b, c).

Recently, deep learning and convolutional neural networks are applied in the domain of iris recognition. Minaee, Abdolrashidiy, and Wang (2017) proposed the deep features extracted from VGG-Net for iris recognition. Authors in this work skipped iris segmentation step in their framework, and hence it can be considered as a peri-ocular recognition more than iris recognition. Gangwar and Joshi (2016) proposed a generalizable iris recognition architecture for iris representation. An open source OSIRIS implementation for iris segmentation is used. While authors note high generalizability and cross-sensor performance, the segmentation errors generated by OSIRIS could be affecting the result of this system. Liu et al.

proposed DeepIris for heterogeneous iris verification (Liu, Zhang, Li, Sun, & Tan, 2016). Also, deep learning based approaches for spoof and contact lens detection can be found in Menotti et al. (2015) and Silva et al. (2015)

1.2.3. CNN for iris segmentation

Li et al. produced two CNN based models for iris segmentation (Liu, Li et al., 2016)–(i) hierarchical convolutional neural network (HCNN) with three blocks of alternative convolutional and pooling layers fed directly in to a fully connected layer; and (ii) multi-scale fully convolutional network (MFCN) which contains six blocks of interconnected alternative Conv and Pool layers fused through a single multiplication layer followed by a Softmax layer. Jalilian and Uhl (2017) proposed three types of fully convolutional encoder–decoder networks for iris segmentation. Arsalan et al. (2017) proposed a two-stage iris segmentation based on CNNs for images captured in visible light. Authors used circular Hough transform to detect rough iris boundary in the first stage. A pre-trained VGG–face model is used in the second stage for the fine adjustment of rough iris boundary obtained in the first stage. In order to overcome the requirement of large labeled data in the approaches mentioned above, Jalilia, Uhl and Kwitt proposed a domain adaptation technique for CNN based iris segmentation (Jalilian, Uhl, & Kwitt, 2017).

1.2.4. Foundation methods

The two primary contribution of this work are (i) a novel iris database augmentation and (ii) a semi parallel deep neural network.

1.2.4.1. Database augmentation. Since deep learning approaches need a large number of samples to train a deep network, data augmentation becomes a crucial step in the training process. Database augmentation is the process of adding variation to the samples in order to expand the database and inject uncertainty to the training set which help the network avoid overfitting and also generalizing the results. Also, the augmentation step can introduce more variations into the database and helps the network to generalize its results. The most well-known augmentation techniques include flipping, rotating and adding distortions to the image are widely used in expanding databases. Such techniques are usually used blindly and do not always guarantee any boost in the performance (Lemley et al., 2017a, b, c).

In Lemley et al. (2017a, b, c), the authors proposed a smart augmentation method which combines two or more samples of the same class and generates a new sample from that class. This method can give superior results compared to classical augmentation techniques. Unfortunately, this method is only applicable to classification problems, and unlike some types of augmentation it cannot be used to manipulate the network results toward pre-determined outcome or task. The ability to do this is another important use of the augmentation, for example adding motion blur to the samples can introduce the robustness to the motion blur in the final results.

The data augmentation technique employed in this work is designed for the specific task of iris recognition. A high quality, ISO standard compliant iris image (Biometrics, 2007) is degraded to make a reasonable representation of the real-world, low-quality consumer grade iris images. This degradation reduces iris–pupil and iris sclera contrast along with the introduction of various noises. Iris maps obtained from state of the art iris segmentation techniques are used to aid this process. This particular strategy of augmentation is employed to harness the highly accurate segmentation capabilities of the state of the art iris segmentor. In this way, images which are a reliable representation of the low-quality consumer images can be obtained along with the corresponding iris map for the training of the neural network.

1.2.4.2. Semi Parallel Deep Neural Network (SPDNN). The second contribution of this work is the use of the recently introduced network design method called Semi Parallel Deep Neural Network (SPDNN) (Bazrafkan & Corcoran, 2017; Bazrafkan, Javidnia, Lemley, & Corcoran, 2017) for generating iris maps from low quality iris images. In an SPDNN, several deep neural networks are merged into a single model to take advantage of every design. For a specific task, one can design several DNN models each of them having advantages and shortcomings. The SPDNN method gives the possibility of merging these networks in layer level using graph theory calculations. This approach maintains the order of the kernels from the parent networks in the merged network. The convergence and generalization of this method along with various application can be found in Bazrafkan and Corcoran (2017) and Bazrafkan, Javidnia et al. (2017). In the present work, the model is trained to generate an iris map from such low-quality images.

1.3. Contribution

This work targets the iris segmentation in low-quality consumer images such as the images obtained from a smartphone. An end to end deep neural network model is proposed to isolate iris region from the eye image. The proposed segmentation technique could be used with any existing state of the art feature extraction and matching module without changing the whole authentication workflow. Performance evaluation of the proposed technique shows advantages over recent iris segmentation techniques presented in the literature. There are notably three primary contributions in this work.

- 1- An improved data augmentation technique optimized to generate diverse low-quality iris images. Such iris images are representative of unconstrained acquisition on a handheld mobile device from multiple established iris research databases.
- 2- A sophisticated iris segmentation network design derived using Semi Parallel Deep Neural Network techniques; Design and optimization methodologies are presented in detail.
- 3- A detailed evaluation of the presented iris segmentation approach is presented on various publically available databases. The presented method is compared with state of the art techniques in iris segmentation.

In the next section, the database and augmentation technique is explained. The network design and Training is explained in Section 3 followed by results given in Section 4. The last section explains the numerical results, experiments on tuning, and comparisons to state of the art segmentation methods.

2. Databases & augmentation methodology

In this work, four datasets are used for training and evaluation. Bath800 (Rakshit, 2007) and CASIA Thousand (“CASIA Iris Image Database”, 2010) have been used in training and testing stages. UBIRIS v2 (Proenca, Filipe, Santos, Oliveira, & Alexandre, 2010) and MobBio (Sequeira, Monteiro, Rebelo, & Oliviera, 2014) are taking part in tuning and also testing. Bath800 and CAISA thousand has been augmented to represent more Real world consumer grade situations. 70% of the samples have been used for training/tuning, 20% for validation and 10% for testing. The network is trained initially on CASIA Thousand and Bath800 and tested on all databases. For further observations, the original network has been tuned on UBIRIS v2 and MobBio separately and also on a mixed UBIRIS+MobBio database. The experiments and discussions are given in Section 5. Following is introducing databases used in this work, followed by the ground truth generation and augmentation.

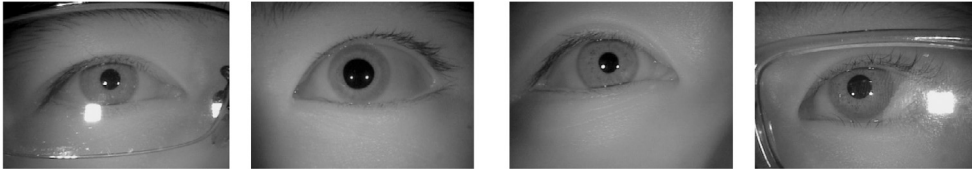


Fig. 1. Eye socket samples from CASIA Thousand database.

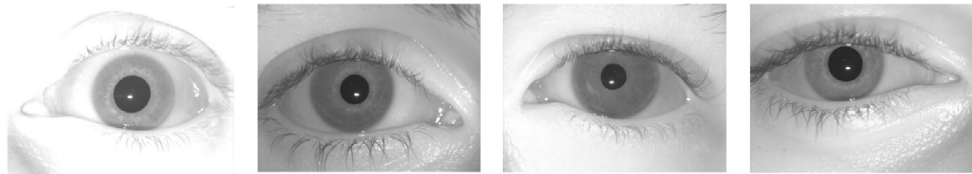


Fig. 2. Eye socket samples from Bath800 database.



Fig. 3. Eye socket samples from UBIRIS database.

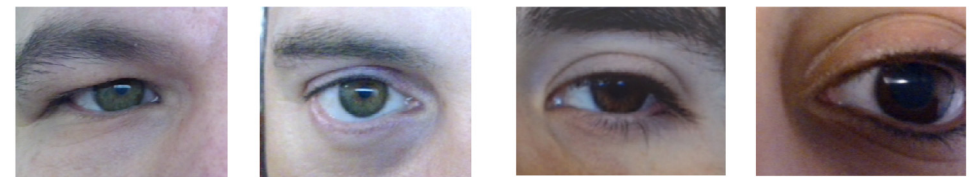


Fig. 4. Eye socket samples from MobBio database.

2.1. Datasets

CASIA Thousand is a subset of CASIA Iris v4 database, and it contains 20000 NIR images captured from 1000 individuals. Images are constrained high quality with high contrast. The resolution is $[640 \times 480]$ for all images. Samples of this database are shown in Fig. 1. Bath800 is a high-quality iris database taken under near infrared illumination using a Pentax C-3516 M with 35 mm lens. Image resolution is $[1280 \times 960]$. The database is made of 31997 images taken from 800 individuals. The images are high quality and high contrast. Fig. 2 shows samples of this database. UBIRIS v2 database has 11102 images taken in visible wavelength with a Canon EOS 5D which are relatively low-quality images captured in an unconstrained environment. The resolution of the database is 400×300 . This database is not used in our training step. Some samples of this database are shown in Fig. 3. MobBio is a multi-modal database including face, iris, and voice of 105 volunteers. The iris subset is used in the current work. The iris images are taken in several orientations, with different levels of occlusion. 16 images are taken from each individual and cropped and resized to resolution 300×200 . This database is highly unconstrained and one of the most challenging sets in iris segmentation/recognition. Some samples of this database are shown in Fig. 4.

2.2. Ground truth generation

2.2.1. Bath800 and CASIA Thousand

Neither of Bath800 and CASIA Thousand databases are provided with ground truth segmentation. As mentioned before these databases contain very high-quality images taken in highly constrained conditions. In this work, the segmentation from high-quality images obtained using a commercial iris segmentation solution (MIRLIN (“MIRLIN”, n.d.)) is considered as the ground truth for training stage. It can be noted that any commercial, high performing segmentation technique could be used here as these high quality images could be segmented accurately using such commercial systems. This specific choice of segmentor used in this work is based on its availability and its performance on large scale iris evaluations (Quinn, Grother, Ngan, & Matey, 2013) Some segmentation examples are given in Figs. 5 and 6. The low resolution segmentation for Bath800 and CASIA Thousand is publicly available.¹

2.2.2. UBIRIS and MobBio

The manual segmentation of UBIRIS is available in IRISSEG-EP database (Hofbauer et al., 2014) generated by WaveLab.² The ground truth generation for this database is not completed. Only segmentations for 2250 images from 50 individuals are given in

¹ <https://Goo.gl/JVksyG>.

² <http://www.wavelab.at/sources/Hofbauer14b/>.

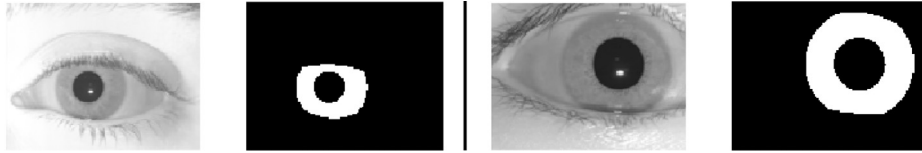


Fig. 5. Bath800 automatic segmentation results.

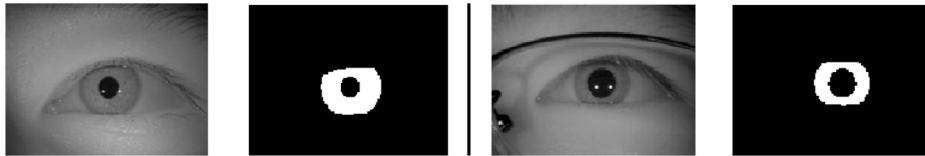


Fig. 6. CASIA Thousand automatic segmentation results.

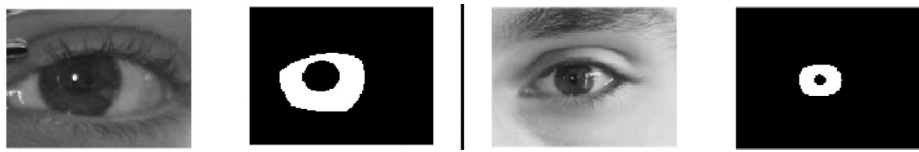


Fig. 7. UBIRIS manual segmentation samples in IRISSEG-EP.

IRISSEG-EP. Some samples for UBIRIS segmentation is shown in Fig. 7.

The manual segmentation for the MobBio database is included in IRISSEG-CC database (Hofbauer et al., 2014) generated by ISLAB.³

The whole MobBio dataset has been segmented in IRISSEG-CC. Some samples for MobBio segmentation are given in Fig. 8.

2.3. Data augmentation

In this work two high-quality databases, Bath800 and CASIA Thousand have been used in training stage. In this section, the augmentations applied to the high-quality iris images are explained. In order to find the best augmentations for the iris images, precise observations have been done on low-quality iris images. The difference between a high quality constrained iris images and consumer grade images depend on five different independent factors: 1- eye socket resolution, 2- image contrast, 3- shadows in the image, 4- image blurring, 5- noise level (Thavalengal et al., 2015a, b). In our observations, the noise level was low in unconstrained images. Noise is a well-studied phenomenon, and image de-noising can be done outside the network and also note that introducing high frequency noise into the dataset trains a low-pass filter inside the network; apply de-noising outside the network gives a higher chance to use the whole network potential to perform the segmentation task. In addition, introducing Gaussian noise into the dataset will cause underfitting as explained in Zheng, Song, Leung, and Goodfellow (2016). Therefore, in this work, the focus is on the first four factors.

The augmentations are applied in a random manner. For example, the contrast, shadow and blurring are applied with random parameters. These operations are lossy. i.e., there is no inverse operation which can regenerate original high quality images from low quality ones. Reducing the resolution of the iris image is a non-reversible task. The contrast reduction contracts a large portion of the histogram into a smaller region, and the quantization process turning the image into unit8 class results in information loss. Shadowing is reducing the intensity of the pixels and in some

cases turns them into absolute zero intensity. This operation is not reversible as well. The blurring effect is applied in random direction and random intensity, which makes it nontrivial to find the deblurring filter.

All these operations are applied with precise observation on low quality consumer level iris images and the parameters are set to simulate the wild conditions. These are next discussed in turn with details of the augmentation approaches taken for each independent factor. The code for augmenting the database is also available.⁴

2.3.1. Eye socket resolution

The resolution of the eye socket plays an essential role in how much information one can extract from the image. In fact, while dealing with the front camera in a mobile phone, the resolution of the camera is lower than the rear cameras. For example, Bazrafkan, Kar, and Costache (2015) observed that the number of the pixels in the iris region for an image taken by a 5 MP front camera of a typical cell phone from the 45 cm distance was just 30 pixels. In order to simulate the low resolution scenario, the high-quality eye socket images and their corresponding ground truth have been resized using bilinear interpolation into smaller images [128 × 96]. This can help the deep network to train faster as well.

2.3.2. Image contrast

In our observations of low quality iris images taken from handheld devices, the intensity properties of the image inside and outside the iris region were different. In fact in the low-quality image set, the region inside the iris was darker than the same region in high-quality images. There was no specific brightness quality for the regions outside the iris in low-quality images. They could be customarily exposed or strongly bright or very dark. In the low-quality images, the regions outside the iris were suffering from the low amount of contrast. In order to apply this transformation to the high-quality images, the contrast reduction inside and outside of the iris region are targeted with different operations. For outside

³ http://islab.hh.se/mediawiki/Iris_Segmentation_Groundtruth.

⁴ https://github.com/C3Imaging/Deep-Learning-Techniques/blob/Iris_SegNet/DBAugmentation/DBAug.m.



Fig. 8. MobBio manual segmentation samples in IRISSEG-CC.

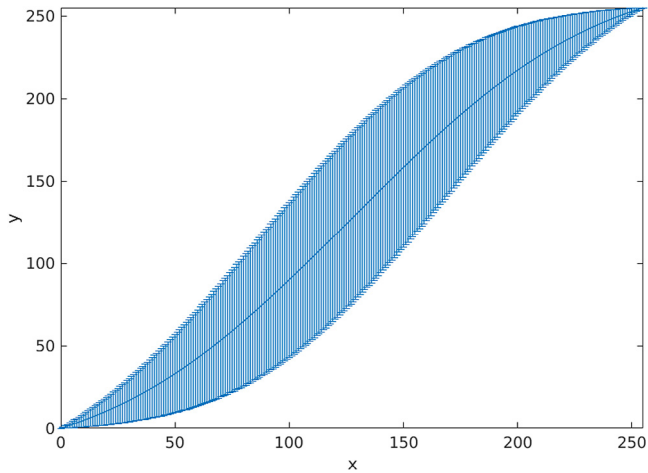


Fig. 9. The histogram mapping for the outside region of the iris.

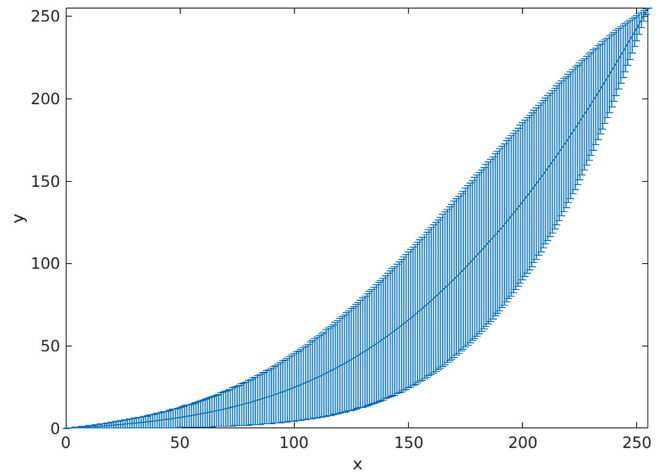


Fig. 10. The histogram mapping for the region inside the iris.

of the iris region the following histogram transformation is applied. This mapping can result in bright, dark, or normally exposed low contrast outputs.

$$y = \text{norm}(\tanh(3 \times (x/255 - 0.5) + \mathcal{U}(-0.3, 0.3))) \times 255 \quad (1)$$

where x is the input intensity in the range $[0,255]$, y is the output intensity in the same range, $\mathcal{U}(a, b)$ is the Uniform distribution between a and b , and the norm function normalize the output between 0 and 1. The Uniform distribution injects an amount of uncertainty into the mapping which helps the generalization of the model. The mean and standard deviation of the histogram mapping curve is shown in Fig. 9.

The inside of the iris is mapped using following histogram transformation.

$$y = \text{norm}(\tanh(3 \times (x/255 - 0.5) - \mathcal{U}(0, 0.8))) \times 255 \quad (2)$$

The mean and standard deviation of the histogram curve is shown in Fig. 10. This transformation is darkening the area while decreasing the contrast at the same time.

An example of applied histogram mapping is showed in Fig. 11.

2.3.3. Shadows in the image

Low quality unconstrained iris images are profoundly altered by the direction of the illumination. In this work, shadowing is carried out by multiplying the image columns by the following function.

$$y = \text{norm}(\tanh(2 \times \text{randSign} \times (x - 0.5 + \mathcal{U}(-0.3, 0.3)))) + \mathcal{U}(0, 0.1) \quad (3)$$

where x is the dummy variable for image column number and y is the coefficient for intensity, $\mathcal{U}(a, b)$ is the Uniform distribution between a and b , and the norm function normalize the output between 0 and 1 and randSign generates a random coefficient in the set $\{-1, 1\}$. The mean and standard deviation of this function

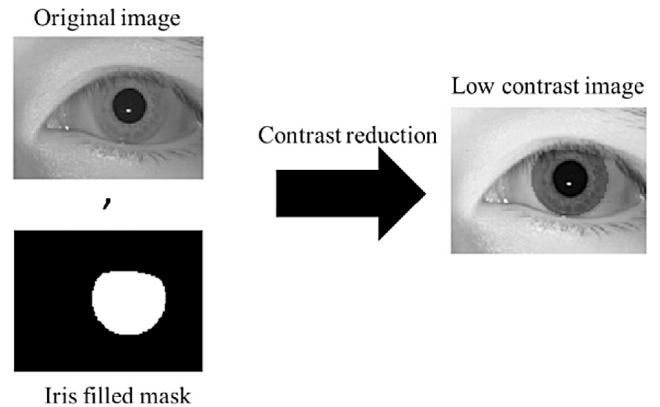


Fig. 11. For inside the iris region, the contrast is reduced, and the region is getting darker. The outside of iris is just altered by decreasing the contrast.

is shown in Fig. 12. Based on the value of the randSign function, the shadowing direction is changed.

An example of shadowing is given in Fig. 13.

2.3.4. Image blurring

Bokeh effect caused by camera focus (Phillips & Komogortsev, 2011), the hand jitter and unwanted head and hand movements are highly degrading the iris image quality in handheld devices. One of the main challenge in iris segmentation in this scenario is the blurring of the iris edges which affects the edge detection task. In majority of the iris segmentation methods, the gradient of the pixel intensity is used to find the iris region, and the blurring is highly altering the gradient quality. The deep neural network is able to solve this problem if enough variation of motion blurring is provided in the dataset. In order to include this effect in the training set, shadowed image is passed through a motion blur filter applying the linear camera motion by $\mathcal{U}(5, 10)$ pixels in the direction $\mathcal{U}(-\pi, \pi)$, where $\mathcal{U}(a, b)$ is the Uniform distribution

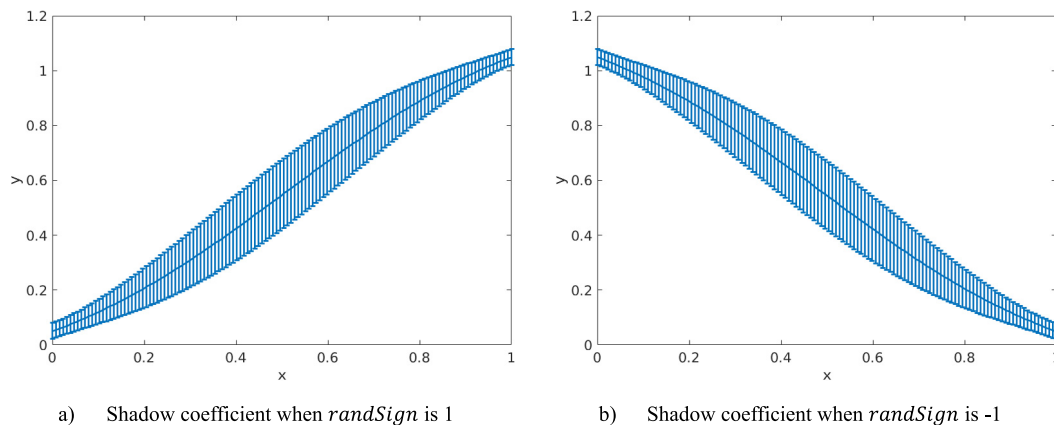


Fig. 12. Mean and standard deviation for shadow coefficients.

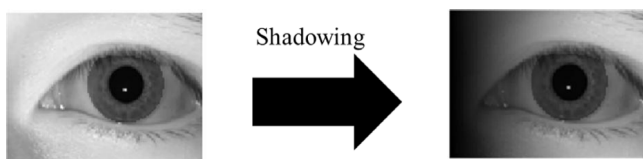


Fig. 13. Shadowing applied to low contrast image.

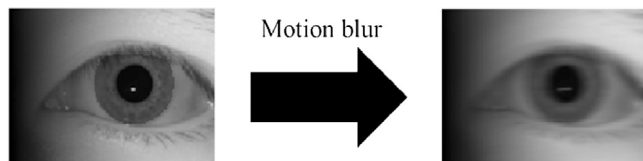


Fig. 14. Applying motion blur in a random direction to the low contrast shadowed image.

between *a* and *b*. The final image after applying motion blur is shown in Fig. 14.

All the samples in Bath800 and CASIA Thousand databases are degraded using this augmentation step. Some examples of the low-quality samples and their corresponding ground truth is given in Fig. 15.

3. Network design and training

3.1. Network design

Deep neural networks are capable of solving highly nonlinear and challenging problems. In this work, four different end to end fully convolutional deep neural networks have been proposed to perform the iris segmentation on low quality images. These networks are merged using SPDNN method, and the number of the channels in each layer is selected in a way that the number of the parameters in the proposed network is similar to the SegNet-basic. The network design and calculating the number of channels in each layer are explained in detail in Supplementary Material #1 and #2 respectively. The SPDNN method is merging several deep neural networks in the layer level using graph theory calculation and graph contraction. This approach is preserving the order of the layers from the parent networks. The convergence and generalization of SPDNN is discussed in Bazrafkan and Corcoran (2017) and other applications of this method is given in Bazrafkan, Javidnia et al. (2017).

The parent networks used in this work are fully convolutional networks with different depths and kernel size each designed to extract different levels of details. The network after merging these networks is shown in Fig. 9 of Supplementary Material #1. The model looks like a U-net (Ronneberger, Fischer, & Brox, 2015) with the difference that there is no pooling applied in our proposed network. The number of the channels in each layer is determined in Supplementary Material #2. The calculations guarantee that the number of the parameters in the presented method is similar to SegNet-Basic proposed in Badrinarayanan, Kendall, and Cipolla (2015). Having the same number of parameters helps to obtain a fair comparison between SegNet-Basic and proposed model.

3.2. Training

The proposed network is an end to end design which means that it accepts the eye socket image and gives the iris map. The network has been trained using lasagna library (Dieleman et al., 2015) on the top of the theano library (Al-Rfou et al., 2016) in python. The loss function used in our work is the mean binary cross-entropy between the output and target given by

$$L = \frac{1}{M \times N \times B} \sum_{k=1}^B \sum_{j=1}^N \sum_{i=1}^M t_{ij} \log(p_{ij}) - (1 - t_{ij}) \log(1 - p_{ij}) \quad (4)$$

wherein t_{ij} is the value of pixel (i, j) in the target image, p_{ij} is the value of pixel (i, j) in the output image for the image of the size $M \times N$ and B is the batch size. The stochastic gradient descent with momentum has been used to update the network parameters. The momentum term prevents the gradient descent to stick in the local minimums, and also speeds up the convergence. In this approach, the gradient decent uses the update value of the previous iteration as the momentum in the current iteration. Suppose the loss function is $L(w)$ where w , is the set of network parameters. The stochastic gradient method with momentum is given by

$$w := w - \eta \nabla L(w) + \alpha \Delta w \quad (5)$$

wherein Δw , is the update in the previous iteration, $\nabla L(w)$ is the gradient value in the current iteration, η is the learning rate and, α is the momentum. In our training experiments, the learning rate and momentum are set to 0.001 and 0.9 respectively. The training method and learning parameters in training the proposed network and SegNet-basic are same.

The network is trained on an augmented version of Bath800 and CASIA1000 originally. Some experiments have been conducted on this original network given in Sections 5.2 and 5.3. These databases are NIR databases. In order to provide a network, segmenting

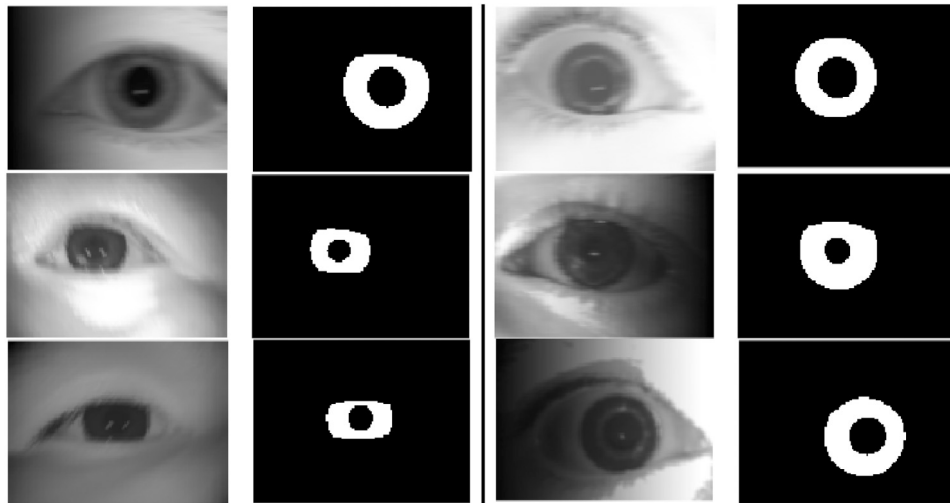


Fig. 15. Augmented samples and their corresponding segmentation map.

visible images, the original network has been tuned on UBIRIS and MobBio databases. The same training method has been used in the tuning stage while learning rate and momentum are set to 0.001 and 0.9 respectively. The train/tune has been done for 1000 epochs. More details on tuning results is given in Supplementary Material #3 and #4.

4. Results

In the test step, each eye socket image is given to the trained/tuned network, and the forward propagation is performed for this input. In the training stage, the output of the network is forced to converge to the iris segmentation map which is a binary image. The output of the network is a grayscale segmentation map, and the binary map is produced by thresholding technique, i.e., the values bigger than a threshold are shifted to 1 and the others to 0. The threshold value 0.45 has been used in our experiments. The output of the proposed model for different databases are shown in Figs. 16 to 19.

Figs. 16 and 17, show the high-quality output for Bath800 and CASIA1000 databases. These datasets are high quality constrained NIR sets, and their images follow a specific distribution which makes it easier for the DNN to perform the segmentation task. Figs. 18 and 19, show the output of the proposed network for more difficult unconstrained UBIRIS and MobBio databases. These two figures show the results of the network tuned on these databases. The results are not as good as Bath800 and CASIA1000, but one should note that these datasets are quite challenging and difficult to segment. The numerical results are given in the following section.

5. Evaluations

Several metrics have been used to evaluate the network and investigate the tuning effect on the segmentation results. These metrics are presented in Table 1. In all equations True Positive is abbreviated as TP, True Negative as TN, False Positive as FP and False Negative as FN. Letter P stands for the number of all Positive cases which is equal to TP+FN and N is the total number of negative cases equals to FP+TN.

5.1. Experimental results

Three main experiments have been conducted to investigate the performance of the proposed network and the effect of the tuning on the results. These experiments are as follows:

- 1- Test on the original network: The proposed network is initially trained on the augmented version of the Bath800 and CASIA1000 databases. The first experiment compares the output of this network for different databases. The test set of Bath800 and CASIA1000 and all the samples of UBIRIS and MobBio are used in the test stage. Section 5.2 discusses this experiment in detail.
- 2- Comparison with SegNet-Basic: This experiment discusses the results of presented network compared the SegNet basic. Training and testing for SegNet-Basic is done similar to the proposed network. This experiment is presented in Section 5.3.
- 3- Comparison to state of the art: In this experiment, the best results of the proposed method is compared with other methods in the literature. The numerical results are presented in Section 5.4.

Two side experiments has been conducted on tuning the network with visible datasets. These experiments are as follows:

- 1- Tuning; Network experiment: In this experiment, the original network trained on the augmented version of Bath800 and CASIA1000 is tuned on UBIRIS and MobBio individually and also on a mixture of these two databases. In this way, the effectiveness of each database in boosting the performance is investigated. The network tuned on each database is tested on all databases. The results and discussions of this experiment is presented in Supplementary Material #3.
- 2- Tuning; Database experiment: This experiment is looking at the results of previous experiment based on each database. There are four networks trained and tuned which are presented in experiment 1 and 4 as follows: (i) Initially trained on Bath800 and CASIA1000. (ii) Tuned on UBIRIS. (iii) Tuned on MobBio. (iv) Tuned on UBIRIS+MobBio. The output of each of these networks for each database and also the average performance is investigated in this experiment. Supplementary Material #4 is presenting this experiment in more detail.

In all experiments, μ stands for the average value for the given measure and σ is its standard deviation over all outputs.

5.2. Test on the original network

In this experiment, the proposed trained network is tested over four databases (Bath800, CASIA1000, UBIRIS, and MobBio). The reason for adding two more databases in the testing procedure is

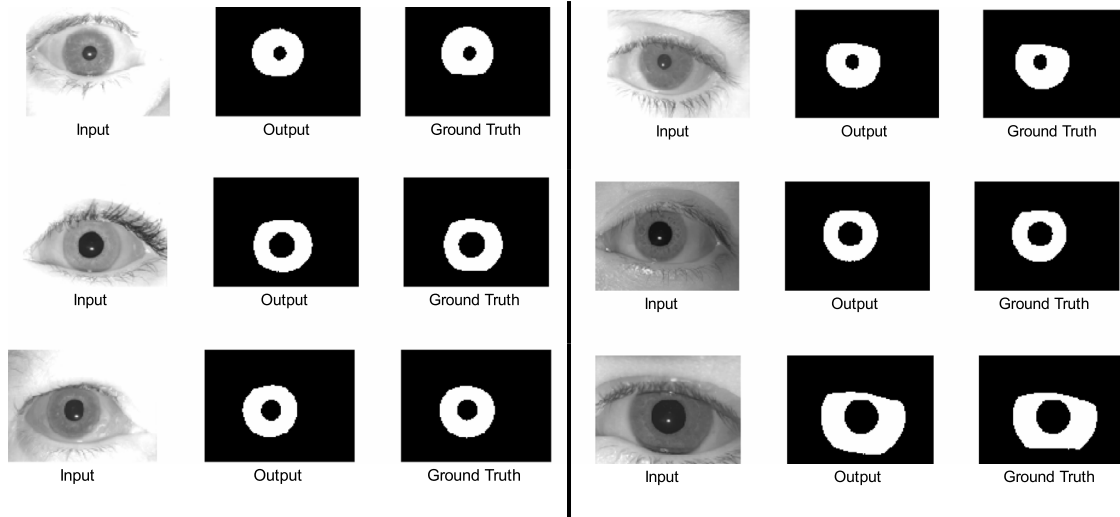


Fig. 16. Output of the network for Bath800 test set. The results show high-quality output in this database.

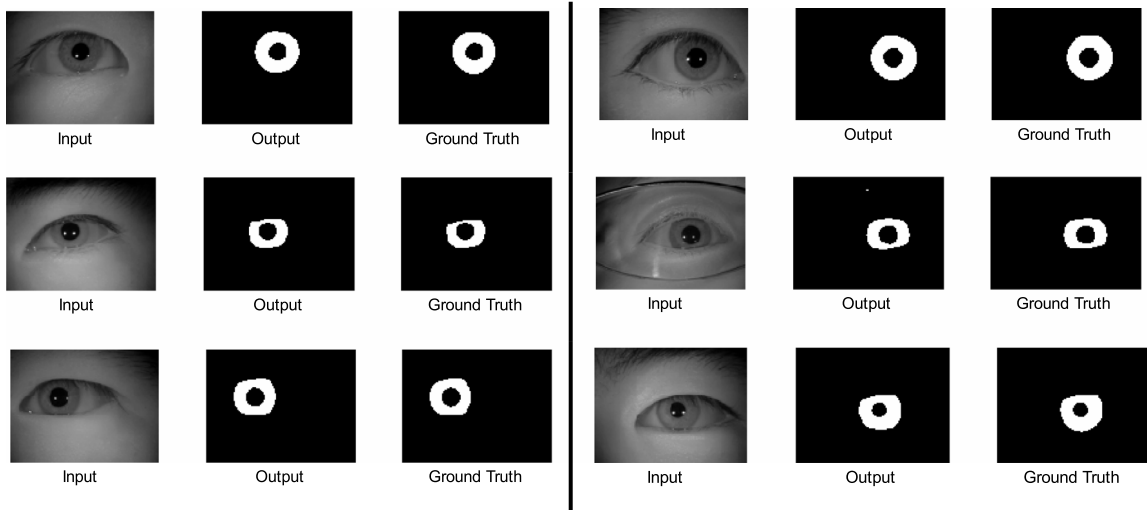


Fig. 17. Output of the network for CASIA1000 test set. The results show high-quality output in this database.

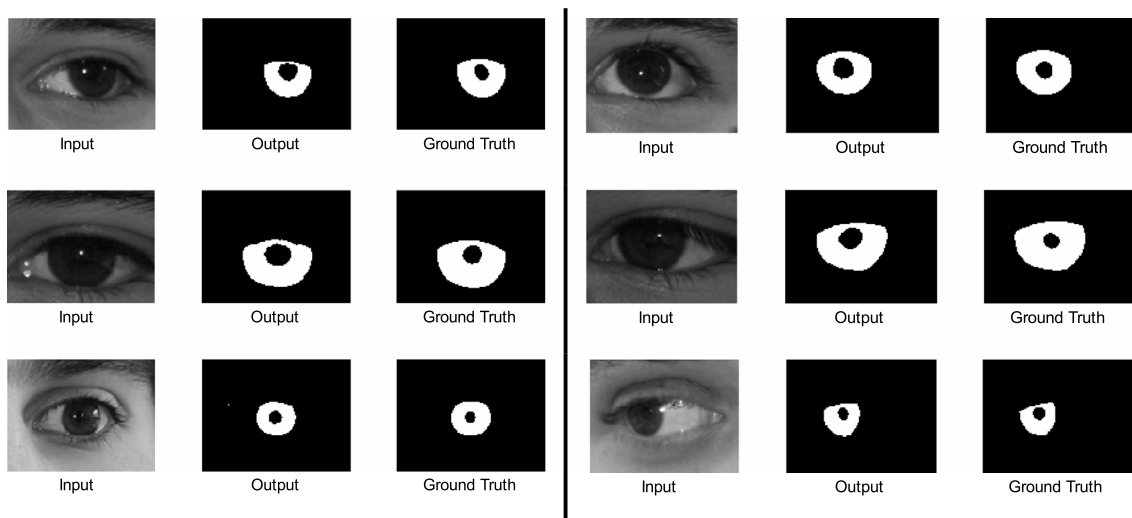


Fig. 18. Output of the network for UBIRIS test set.

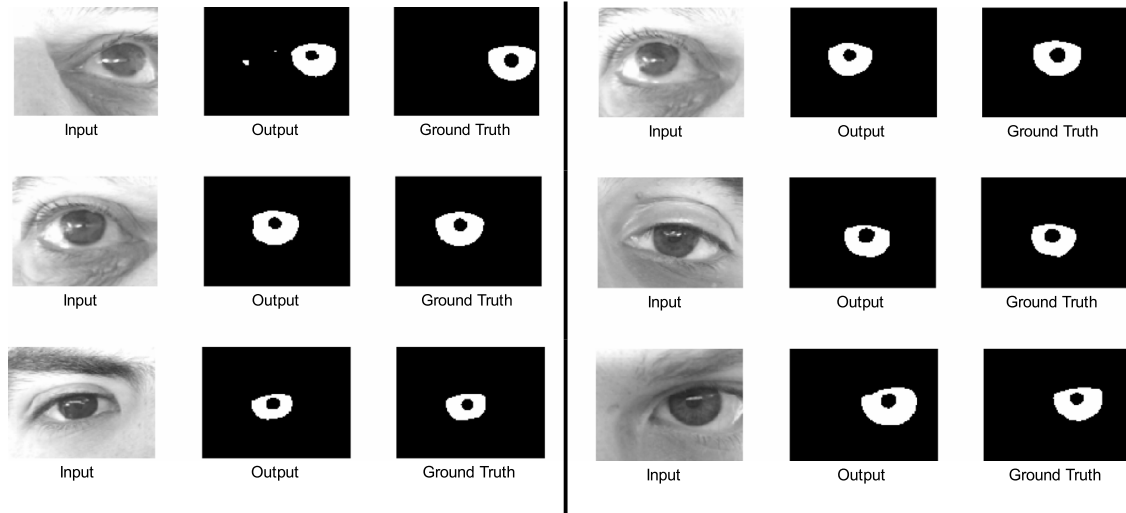


Fig. 19. Output of the network for MobBio test set.

Table 1

Metrics used in the evaluation section.

Measure	Description
Accuracy	Accuracy represents the ratio of all true results divided by the number of all samples given by $\text{Accuracy} = \frac{TP+TN}{P+N}$
Sensitivity or True Positive Rate (TPR)	This measure indicates the ability of the model to recall true positive over all positive samples. i.e., a model with high sensitivity can rule out negative samples more efficiently. Sensitivity is given by $\text{Sensitivity} = \frac{TP}{TP+FN} = \frac{TP}{P}$
Specificity or True Negative Rate	This measure indicates the ability of the model to recall true negative over all negative samples. i.e., a model with high specificity can find positive samples more efficiently. Specificity is given by $\text{Specificity} = \frac{TN}{FP+TN} = \frac{TN}{N}$
Precision or Positive Predictive Value (PPV):	Precision is the probability that the positive output is true positive in the space of all positive outcomes, which is given by $\text{Precision} = \frac{TP}{TP+FP}$
Negative Prediction Value (NPV)	NPV is the probability that the negative output is true negative in the space of all negative outcomes, which is given by $\text{NPV} = \frac{TN}{TN+FN}$
F1 score	This measures the ability of the model to recall true positive cases and at the same time not missing positive cases. F1 score is given by $\text{F1 score} = \frac{2TP}{2TP+FP+FN}$
Matthew Correlation Coefficient (MCC)	Is a metric measuring the quality of binary classifiers. The critical property of MCC is its independence of the size of each class. It varies in the range $(-1,1)$ wherein 1 indicates the perfect model, and -1 declared that all output values are the inverse of the target value. MCC is given by $\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
Informedness	Is a metric showing the probability of informed decision given by the model. It ranges from -1 to 1 ; where 0 shows a random decision and 1 indicates no false outputs. Informedness is given by $\text{Informedness} = \frac{TP}{TP+FN} + \frac{TN}{TN+FP} - 1$
Markedness	Is a measure of the information content and information value of the model's output (Battistella, 1990). Markedness is given by $\text{Markedness} = \frac{TP}{TP+FP} + \frac{TN}{TN+FN} - 1$

to observe the ability of the network in generalizing over other databases. One of the main concerns in DNN community is to be able to generalize the trained network to unconstrained environments. This is happening since the majority of Machine learning schemes are sharing the same database in train and test stage. The neural networks learn the distribution of the data for the given database, and since the test set follows the same distribution, it gives promising results on the test set. Bazrafkan, Nedelcu, Filipczuk, and Corcoran (2017) discusses this problem in more detail.

Since the network is trained on a merged version of the Bath800 and CASIA1000; only the test sub-set of these databases has been used in testing stage. However, the network never saw the UBIRIS and MobBio set before. Therefore all samples of these databases have been used for testing. The numerical results are shown in Table 2. From this experiment, the network gives better results for Bath800 and CASIA1000 which is expected.

The network has already observed these two databases in training stage and learned their distribution which justifies the

Table 2

Testing on the original network. Metrics measured for different databases. Green means higher performance and red declares lower quality results. A higher value of μ and lower value for σ are desirable.

		Bath800	CASIA1000	UBIRIS	MobBio
Accuracy	μ	98.55%	99.71%	97.82%	96.12%
	σ	1.43%	0.33%	1.49%	3.16%
Sensitivity	μ	96.03%	97.96%	74.29%	65.79%
	σ	4.76%	2.95%	20.12%	23.94%
Specificity	μ	99.10%	99.82%	99.25%	97.96%
	σ	1.07%	0.20%	1.08%	2.19%
Precision	μ	96.05%	97.13%	85.65%	68.71%
	σ	4.46%	3.10%	19.94%	27.88%
NPV	μ	99.05%	99.87%	98.40%	97.91%
	σ	1.49%	0.28%	1.20%	1.73%
F1-Score	μ	95.93%	97.50%	78.08%	65.96%
	σ	3.88%	2.51%	19.79%	25.06%
MCC	μ	0.951	0.9737	0.7792	0.647
	σ	0.0421	0.025	0.1909	0.2611
Informedness	μ	0.9514	0.9779	0.7354	0.6375
	σ	0.0486	0.0297	0.2028	0.2519
Markedness	μ	0.951	0.97	0.8406	0.6663
	σ	0.0468	0.0312	0.1996	0.2891

higher performance on these databases. Having a lower value of sensitivity and precision in UBIRIS and MobBio databases declares the amount of uncertainty of the model in giving back the positive cases. Moreover, the high value of specificity and NPV shows that the trained model was able to rule out non-iris pixels in all databases. The value of F1-score, MCC, Informedness, and Markedness is high for Bath800 and CASIA1000 which indicate the ability of the network to produce consistent segmentations both in finding iris and non-iris pixels for these databases. The same measures return average values for UBIRIS database. This means that the network is generalized for semi-wild environments. Moreover, the low value for MobBio indicates that the network is not much reliable to work in consumer level environments. In general, the presented network is reliable in returning non-iris pixels in challenging scenarios. Note that both UBIRIS and MobBio are visible databases and that MobBio is a very challenging database. In Section 5.4.2 the presented network is compared to other methods on the MobBio database.

5.3. Comparison with SegNet-basic

SegNet (Badrinarayanan et al., 2015) is one of the most successful DNN approaches in semantic segmentation. SegNet-basic is the small counterpart of the original SegNet. As explained in Supplementary Material #2, our proposed architecture contains almost the same number of parameters as SegNet-basic. This gives us the opportunity to conduct fair comparisons between two models. The original SegNet-basic is trained on the road scenes for object segmentation task. In order to conduct fair comparisons with the proposed design, the SegNet-basic network is tuned for iris segmentation by training it on the same data with same hyper-parameters as our proposed model. Table 3 shows the results for SegNet-basic tested on four databases Bath800, CASIA1000, UBIRIS and MobBio. Note that the network is tested on the test set of Bath800 and CASIA1000 and all samples of UBIRIS and MobBio.

The SegNet-basic and proposed network contain almost the same number of parameters, so these two networks occupy the same amount of memory. The proposed architecture has more weight connections. On a Geforce 750Ti graphic card the processing load for the proposed model was 0.77 Mp/s (Megapixel per

second) which is comparable to the SegNet-basic 0.6144 Mp/s, while the proposed model have considerably better performance.

Results show SegNet-basic has considerably better performance on Bath800 and CASIA1000 than other two databases; which is expectable since the network is trained on former databases. In the following subsections, these results are compared to presented network in order to find the advantages and shortcomings of each design.

5.3.1. Comparing results on Bath800 and CASIA1000

Since both networks are trained on Bath800 and CASIA1000 databases, the numerical test results show the capability of each design in capturing the probability distribution of the training set. Figs. 20 and 21 illustrate the comparisons between the proposed method and SegNet-basic over Bath800 and CASIA100. These figures show the mean value for each metric.

From these figures, it is concluded that the proposed method is giving better results on the test set of Bath800 and CASIA1000. Since these two datasets have been used to train both networks, these comparisons show the higher capacity of the proposed method in learning the data distribution in training stage. At the same time, one can comment that learning the training distribution can be a sign of overfitting. In order to investigate this effect, both networks are tested on two other databases UBIRIS and MobBio explained in the next section.

5.3.2. Comparing results on UBIRIS and MobBio

It is always important to investigate the performance of a model over databases which has not been used in the training stage in order to get better ideas on the model quality in unconstrained environments. In fact, the network is learning the samples which are present in the training set, and a good model should be able to generalize the results for other samples specially unconstrained, consumer graded and difficult ones. In this section, the results for the SegNet-basic network is compared with the proposed network for UBIRIS and MobBio databases (which are not used in the training stage). The results are shown in Fig. 22 for UBIRIS and Fig. 23 for MobBio.

The presented method is providing higher accuracy than SegNet-basic which implies the better quality in returning true

Table 3

SegNet-basic. Metrics measured for different databases. Green means better quality and red declares lower quality results. A Higher value of μ and lower value for σ are desirable.

		Bath800	CASIA1000	UBIRIS	MobBio
Accuracy	μ	97.84%	99.36%	94.99%	94.02%
	σ	1.68%	0.46%	2.95%	3.73%
Sensitivity	μ	94.20%	96.22%	85.41%	80.67%
	σ	6.01%	4.98%	15.70%	21.58%
Specificity	μ	98.60%	99.55%	95.71%	94.95%
	σ	1.41%	0.30%	3.01%	3.34%
Precision	μ	94.04%	93.01%	58.60%	51.93%
	σ	5.58%	4.92%	23.84%	21.01%
NPV	μ	98.66%	99.76%	98.90%	98.67%
	σ	1.84%	0.38%	1.27%	1.71%
F1-Score	μ	93.91%	94.45%	66.59%	61.10%
	σ	4.42%	3.99%	20.25%	20.10%
MCC	μ	0.927	0.942	0.6694	0.6092
	σ	0.048	0.0395	0.1873	0.2052
Informedness	μ	0.928	0.9578	0.8113	0.7562
	σ	0.0587	0.0497	0.1622	0.2264
Markedness	μ	0.927	0.9277	0.575	0.506
	σ	0.0561	0.0494	0.2355	0.216

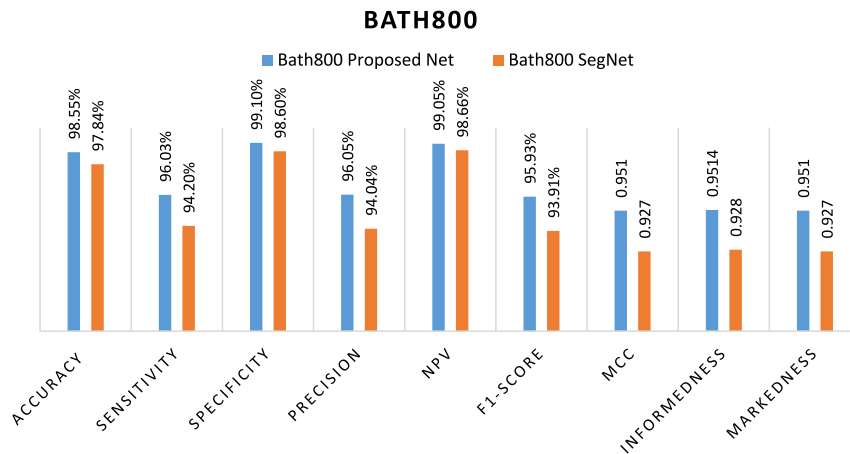


Fig. 20. Comparisons between the presented network and SegNet-basic on Bath800. A higher value indicates better performance.

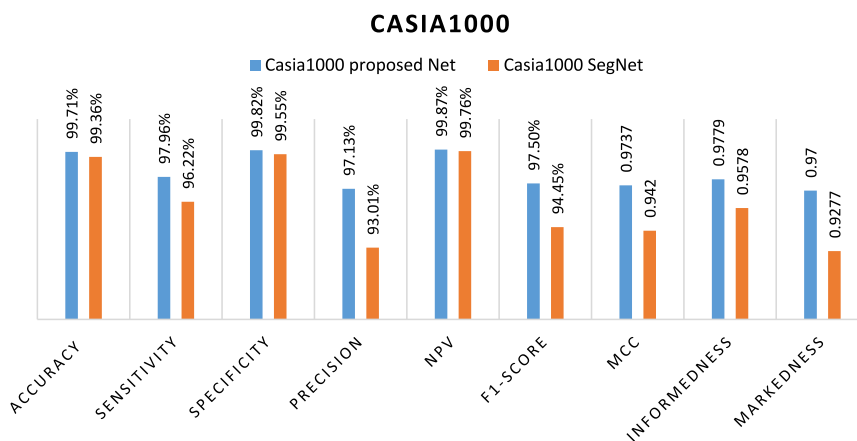


Fig. 21. Comparisons between the presented network and SegNet-basic on CASIA1000. Higher value indicates better performance.

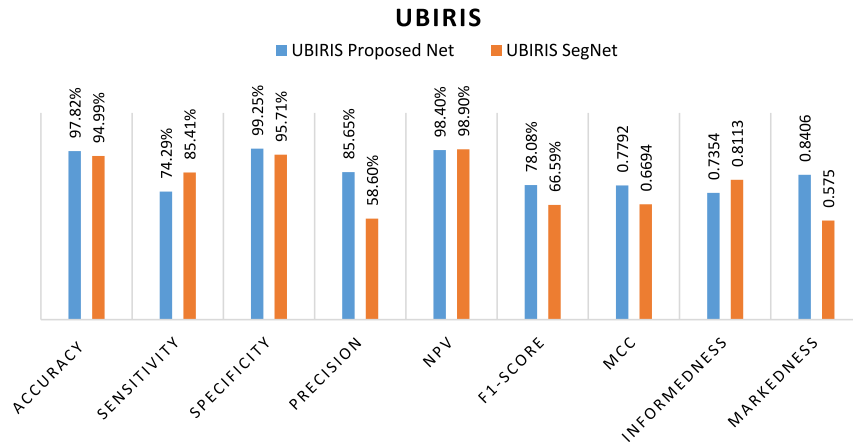


Fig. 22. Comparisons between the presented network and SegNet-basic on UBIRIS. A higher value indicates better performance.

results in the space of all results. This is while SegNet-basic has higher sensitivity and NPV which it means that this architecture is more efficient in ruling out non-iris pixels while the presented model has better performance in finding positive samples due to its higher specificity and precision. Lower FPR shows that the proposed model had lower probability in returning a negative decision and higher FNR shows that SegNet-basic is less probable in making a mistake in returning positive decisions. However, in average the proposed method is more efficient since it has a higher value for F1-score which is the harmonic average of precision and sensitivity. Moreover, also higher MCC shows that the overall performance of the presented network is better than SegNet-basic for UBIRIS dataset.

The numerical results of testing both networks on MobBio dataset are shown in Fig. 23.

Results for MobBio database is in the same direction as UBIRIS. The proposed model got higher accuracy which means it has better performance in finding iris pixels. Values for sensitivity and NPV shows better performance of SegNet-basic in ruling out non-iris pixels. And observations on specificity and precision shows the better performance of the proposed model in returning iris pixels. However, on average the proposed method has better performance due to higher numerical values for F1-score and MCC.

The results of testing both networks on UBIRIS and MobBio datasets demonstrate the overall improved performance of proposed network over SegNet-basic. This shows that the model is not only more capable of learning the training data distribution, but also it has a better ability to generalize to unconstrained, consumer level environments.

5.4. Comparison to state of the art

In this next experiment, the proposed method is compared to the most advanced and state of the art segmentation methods in the literature. In the first part, the accuracy of the proposed method is compared to other methods over UBIRIS database. Moreover, in the second part, the sensitivity, precision, and F1-score of the proposed method is compared with some other methods over UBIRIS, MobBio, and CASIA databases. The results presented here are the best results of our networks after tuning.

5.4.1. Accuracy on the UBIRIS database

The comparisons of the proposed method with the state of art methods over UBIRIS database is illustrated in Fig. 24.

The MFCN and HCNN (Liu, Li et al., 2016) methods are using a 22 layer, deep neural network to perform the iris segmentation. Zhao and Ajay (2015) utilizes the Total Variation (TV) model to

overcome the problem of low contrast and noise interference in the eye socket image. Tan, He, and Sun (2010) proposes an integrodifferential constellation followed by a curvature fitting model to find the iris area. In Radman, Zainal, and Suandi (2017) The Histogram of Oriented Gradients (HOG) is introduced as feature and Support Vector Machine (SVM) is used to perform the automatic segmentation of iris. The random walker algorithm is used to generate the iris map in Tan and Kumar (2013). In Proenca (2010) the sclera and iris regions are detected separately using neural networks as classifiers, and polynomial fitting is applied estimating the final iris region. Tan and Kumar (2012) proposes a post-classification procedure including reflection and shadow removal and several refinements on pupil and eyelid localizations to get higher performance on iris segmentation task. From Fig. 24, the proposed methods gives the best accuracy for UBIRIS database. The main advantage of our work was by selecting the proper data augmentation described in Section 2.3. Augmentation step is essential in any model which is designed to work in unconstrained conditions.

The network is learning the distribution of the train set and therefore, designing a set which is representative of the consumer level conditions is crucial in order to get reasonably high performance, i.e., if one can mimic the real-life situations by introducing enough variations to the training set, it is highly probable that the network is able to generalize the learning into non constrained input test samples. Moreover, also tuning is an essential part of our approach to getting a better result for a pre-defined condition. The original network was trained on Bath800 and CASIA 1000 databases which are NIR iris images. At the same time, the UBIRIS database is not big enough to train a DNN without encountering over-fitting condition. One of the best approached to train a network on such a small database is to transfer the information from the original network (which is trained on NIR images) and tune it on the new database.

The other advantage of the proposed method is the semi parallel design of the network. In this approach, one can take advantage of several architectures at the same time. This is while the information flow inside the network is not limited to a single path but mixing and merging of several paths.

5.4.2. Experiments on sensitivity, precision, and F1-score

The sensitivity, precision, and F1-score of five iris segmentation methods (CAHT, GST, IFFP, Osiris, and WAHET) on several databases including UBIRIS, MobBio and CASIA are given in Hofbauer et al. (2014). In our work, the comparisons with these methods have been conducted on the same databases, and the results are given in Figs. 25 to 27. The results reported by Hofbauer et al.

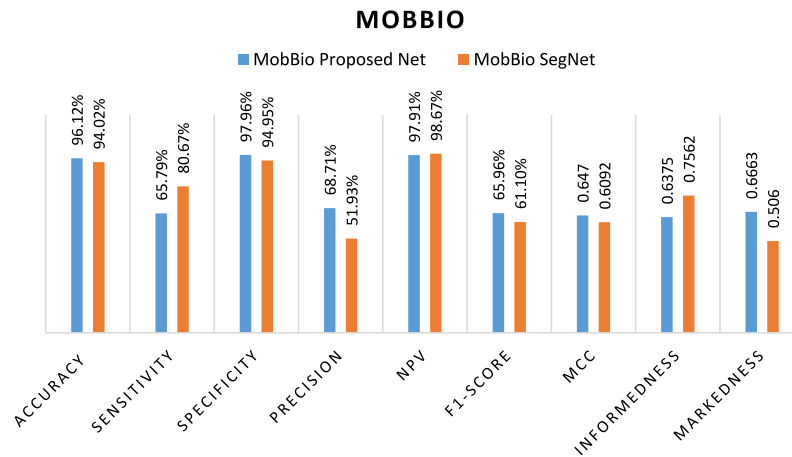


Fig. 23. Comparisons between the presented network and SegNet-basic on MobBio. A higher value indicates better performance.

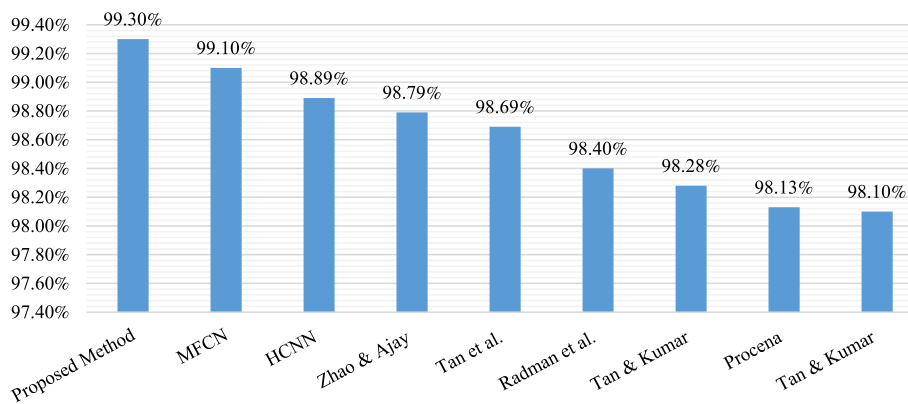


Fig. 24. Accuracy of proposed method vs. other methods over UBIRIS database.

(2014) for UBIRIS database provides a detailed performance metric including sensitivity, precision and f1-score. These metrics for each presented algorithm (CAHT, GST, IFFP, OSIRIS, and WAHET) are calculated by comparing the algorithms results with the ground truth. Hofbauer et al. (2014) work aim to streamline the development and analysis of newer iris segmentation algorithms by providing a detailed baseline result. Hence the results presented there for CAHT, GST, IFFP, OSIRIS, and WAHET segmentation algorithms is expected to be the best results for these methods.

More recent work on UBIRIS, including the segmentation approach based on CNNs such as Liu, Li et al. (2016) provides just the segmentation accuracy. A detailed comparison of segmentation accuracy of these techniques to our proposed technique is shown in Fig. 24. As the algorithms used in these works are not available for research community for reproducing their results, calculating new metrics which are not provided by the authors are not feasible.

Other recent works on iris segmentation such as Abdullah, Dlay, Woo, and Chambers (2017); Ehsaneddin (Jalilian & Uhl, 2017; Morley & Foroosh, 2017) neither compare their results on UBIRIS dataset nor make the segmentation technique public for a fair comparison of their techniques in the other databases.

Sensitivity and precision metrics measure the quality of the network in ruling out non-iris pixels and detection iris ones respectively, and F1-score is the harmonic average of these two metrics. The higher values correspond to better performance. As shown in Figs. 25 to 27, the proposed method gives superior results compared to other approaches on UBIRIS and MobBio databases. Moreover, on the high-quality CASIA database, the proposed method is still giving better results. This shows that the proposed method is

performing on low-quality consumer graded iris images as good as constrained high-quality samples. This is essential while the user tries to capture the iris information in handheld devices where there is hand shaking, sparse illumination and low-quality front cameras. The proposed network shows that this conditions could be compensated by augmenting the data and also merging several designs into a single network and the numerical results show promising performance of the proposed scheme.

6. Conclusions

In this work, a deep neural network framework has been presented to segment low quality, consumer graded iris images.

There are three main contributions in this work.

- (i) The data augmentation, wherein the high quality eye socket images from Bath800 and CASIA1000 database are degraded and manipulated to give a proper approximation of the low quality images. Four different factors have been considered including image resolution, contrast, shadow and motion blurring. The augmented images give a close approximation of low quality unconstrained iris images.
- (ii) The recently introduced Semi Parallel Deep Neural Network method has been used to design a fully convolutional network by mixing and merging four parent networks. Each of these networks take advantage of different kernel sizes and depths which are extracting and processing different feature levels. The final design is similar to U-Net without pooling.

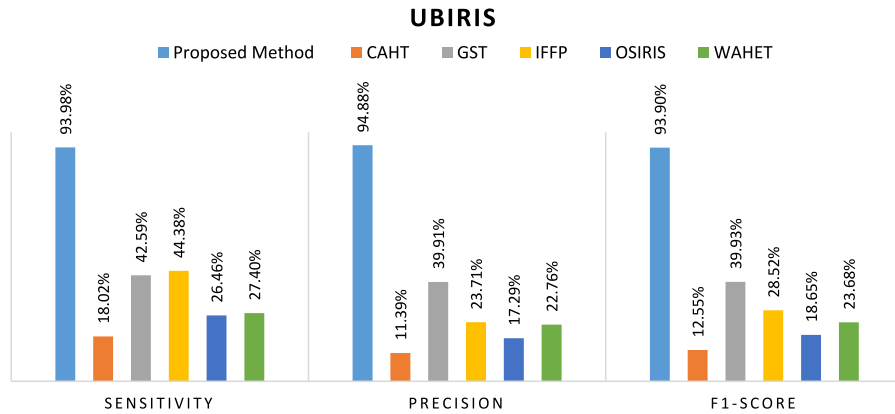


Fig. 25. Sensitivity, Precision, and F1-score on UBIRIS database for proposed method vs. five other methods. Higher values indicate better performance.

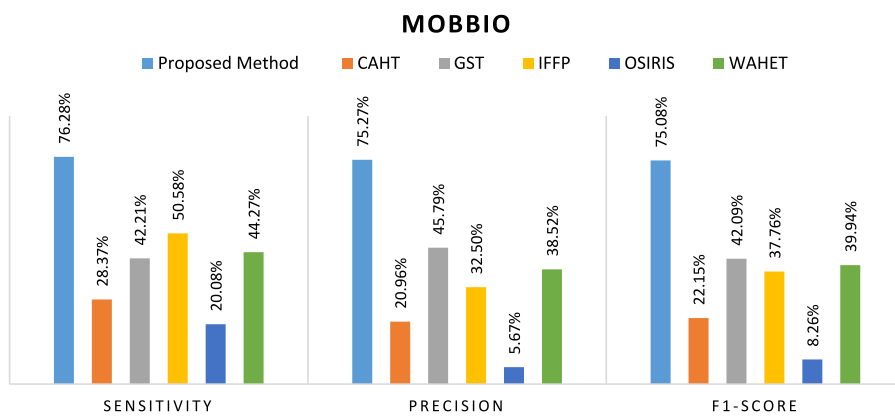


Fig. 26. Sensitivity, Precision, and F1-score on MobBio database for proposed method vs. five other methods. Higher values indicate better performance.

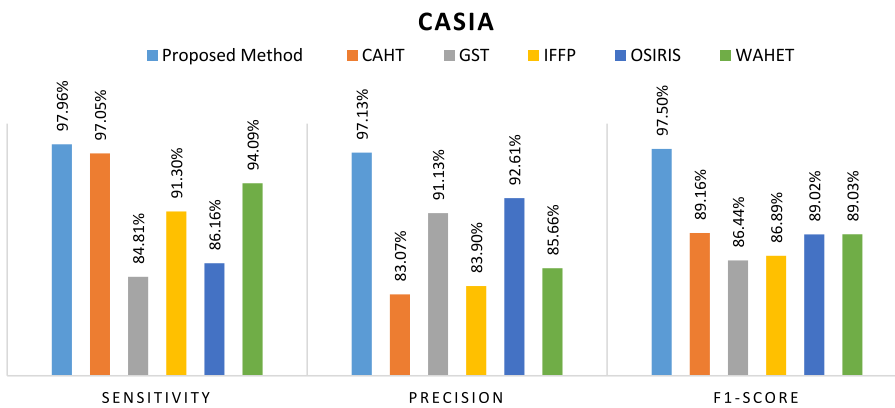


Fig. 27. Sensitivity, Precision, and F1-score on CASIA database for proposed method vs. five other methods. Higher values indicate better performance.

(iii) Inter-database evaluations are giving a more realistic overview of the network performance. Here a very essential problem in deep learning community is addressed wherein the researchers are training a DNN on a specific database and test it on the same database. In this work, every network was tested on Bath800, CASIA1000, UBIRIS, and MobBio. Employing this approach gives a realistic foresight of the performance on real world situations.

The proposed model has been initially trained on the augmented version of the Bath800, and CASIA1000 databases and further experiments were carried out by tuning the original network on UBIRIS and MobBio. Tuned networks were tested on all

databases, and the effect of tuning was widely investigated. Our experiments show that the tuning boosts the performance for the database that the network is tuned on. This is expectable except for databases with very unspecific distributions which will decrease the performance after tuning. Another conclusion is that tuning the model is only advisable when the end use conditions are known. But if the end use conditions are not known, such tuning is not advised.

Since the presented network is a large model which is not easily implementable on a low power handheld hardware, the future works include optimizing the network, training a smaller network, or binarizing the model to reduce the number of calculations and the memory usage. Optimizing the network includes reducing the

parameter precision down to binary or ternary (Li, Zhang, & Liu, 2016). This will give, up to 32x memory compression and also reduces the calculation load extensively by eliminating most of multiplications in the model. Another approach is to design a model with fewer parameters. Currently, our target is to reduce the number of parameters to the rate of 10x without causing considerable cutback in the performance.

Acknowledgments

This research is funded under the SFI Strategic Partnership Program by Science Foundation Ireland (SFI) and FotoNation Ltd. Project ID: 13/SPP/I2868 on Next Generation Imaging for Smartphone and Embedded Platforms.

We gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU used for this research.

Portions of the research in this paper use the CASIA-IrisV4 collected by the Chinese Academy of Sciences' Institute of Automation (CASIA).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.neunet.2018.06.011>.

References

- Abdullah, M. A. M., Dlay, S. S., Woo, W. L., & Chambers, J. A. (2017). Robust iris segmentation method based on a new active contour force with a noncircular normalization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(12), 3128–3141.
- Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., & Ballas, N., et al., (2016). Theano: A [Python] framework for fast computation of mathematical expressions. arXiv E-Prints, abs/1605.0. Retrieved from <http://arxiv.org/abs/1605.02688>.
- Alonso-Fernandez, F., & Bigun, J. (2013). Quality factors affecting iris segmentation and matching. In *Proceedings - 2013 international conference on biometrics*, <http://dx.doi.org/10.1109/ICB.2013.6613016>.
- Arsalan, M., Hong, H. G., Naqvi, R. A., Lee, M. B., Kim, M. C., Kim, D. S., et al. (2017). Deep learning-based iris segmentation for iris recognition in visible light environment. *Symmetry*, 9(11).
- Badrinarayanan, V., Kendall, A., & Cipolla, R. (2015). SegNet: [A] Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. CoRR, abs/1511.0. Retrieved from <http://arxiv.org/abs/1511.00561>.
- Battistella, E. L. (1990). *Markedness: The evaluative superstructure of language*. SUNY Press.
- Bazrafkan, S., & Corcoran, P. (2017). Semi-Parallel Deep Neural Networks (SPDNN), Convergence and Generalization. Retrieved from <http://arxiv.org/abs/1711.01963>.
- Bazrafkan, S., Javidnia, H., Lemley, J., & Corcoran, P. (2017). Depth from Monocular Images using a Semi-Parallel Deep Neural Network (SPDNN) Hybrid Architecture. arXiv Preprint arXiv:1703.03867 (pp. 1–15). Retrieved from <http://arxiv.org/abs/1703.03867>.
- Bazrafkan, S., Kar, A., & Costache, C. (2015). Eye gaze for consumer electronics: Controlling and commanding intelligent systems. *IEEE Consumer Electronics Magazine*, 4(4), 65–71. <http://dx.doi.org/10.1109/MCE.2015.2464852>.
- Bazrafkan, S., Nedelcu, T., Filipczuk, P., & Corcoran, P. (2017). Deep learning for facial expression recognition: a step closer to a smartphone that knows your moods. In *IEEE international conference on consumer electronics*.
- Bigun, J., Alonso-Fernandez, F., Hofbauer, H., & Uhl, A. (2016). Experimental analysis regarding the influence of iris segmentation on the recognition rate. *IET Biometrics*, 5(3), 200–211. <http://dx.doi.org/10.1049/iet-bmt.2015.0069>.
- Biometrics, I. S. O. J. S. (2007). ISO 29794-1 biometric sample quality. Committee Draft, 1.
- Bowyer, K. W., Hollingsworth, K., & Flynn, P. J. (2008). Image understanding for iris biometrics: A survey. *Computer Vision and Image Understanding*, 110, 281–307. <http://dx.doi.org/10.1016/j.cviu.2007.08.005>.
- Bowyer, K. W., Hollingsworth, K. P., & Flynn, P. J. (2013). A survey of iris biometrics research: 2008–2010. *Handbook of Iris Recognition*, (August), 15–54. http://dx.doi.org/10.1007/978-1-4471-4402-1_2.
- CASIA Iris Image Database, (2010). Retrieved July 24, 2017, from <http://biometrics.idealtest.org/>.
- Corcoran, P., & Costache, C. (2016). Smartphones, Biometrics, and a Brave New World. *IEEE Technology and Society Magazine*, 35(3), 59–66. <http://dx.doi.org/10.1109/MTS.2016.2593266>.
- Daugman, J. G. (1994, March). Biometric personal identification system based on iris analysis. Google Patents.
- Daugman, J. (2004). How iris recognition works. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1), 21–30. <http://dx.doi.org/10.1109/TCSVT.2003.818350>.
- Daugman, J. (2007). New methods in iris recognition. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics: A Publication of the IEEE Systems, Man, and Cybernetics Society*, 37(5), 1167–1175.
- De Marsico, M., Nappi, M., & Proença, H. (2017). Results from MICHE II—mobile iris challenge evaluation II. *Pattern Recognition Letters*, 91, 3–10. <http://dx.doi.org/10.1016/j.patrec.2016.12.013>.
- De Marsico, M., Nappi, M., Riccio, D., & Wechsler, H. (2015). Mobile iris challenge evaluation (MICHE)-I, biometric iris dataset and protocols. *Pattern Recognition Letters*, 57, 17–23. <http://dx.doi.org/10.1016/j.patrec.2015.02.009>.
- Dieleman, S., Schluter, J., Raffel, C., Olson, E., Sonderby, S. K., & Nouri, D. et al., (2015, August). Lasagne: First release. <http://dx.doi.org/10.5281/zenodo.27878>.
- Erbilek, M., Da Costa-Abreu, M. C., & Fairhurst, M. (2012). Optimal configuration strategies for iris recognition processing. In *IET conference on image processing* (pp. B2–B2). <http://dx.doi.org/10.1049/cp.2012.0451>.
- Gangwar, A., & Joshi, A. (2016). DeepIrisNet: Deep iris representation with applications in iris recognition and cross-sensor iris recognition. In *Proceedings - international conference on image processing*, Vol. 2016–August, (pp. 2301–2305). <http://dx.doi.org/10.1109/ICIP.2016.7532769>.
- Gangwar, A., Joshi, A., Singh, A., Alonso-Fernandez, F., & Bigun, J. (2016). IrisSeg: A fast and robust iris segmentation framework for non-ideal iris images. In *2016 international conference on biometrics* <http://dx.doi.org/10.1109/ICB.2016.7550096>.
- Haindl, M., & Krupička, M. (2015). Unsupervised detection of non-iris occlusions. *Pattern Recognition Letters*, 57, 60–65. <http://dx.doi.org/10.1016/j.patrec.2015.02.012>.
- He, Z., Tan, T., Sun, Z., & Qiu, X. (2009). Toward accurate and fast iris segmentation for iris biometrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(9), 1670–1684. <http://dx.doi.org/10.1109/TPAMI.2008.183>.
- Hofbauer, H., Alonso-Fernandez, F., Wild, P., Bigun, J., & Uhl, A. (2014). A ground truth for Iris segmentation. In *Proceedings - international conference on pattern recognition* (pp. 527–532). <http://dx.doi.org/10.1109/ICPR.2014.101>.
- Ioffe, S., & Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456).
- Irsch, K., Guyton, D. L., & Johns, H. M. (2009). Anatomy of eyes. *Encyclopedia of Biometrics*, (April), 1212–1217. http://dx.doi.org/10.1007/978-0-387-73003-5_257.
- Jalilian, E., & Uhl, A. (2017). Iris segmentation using fully convolutional encoder-decoder networks. *Advances in Computer Vision and Pattern Recognition, Part F1*, 133–155. http://dx.doi.org/10.1007/978-3-319-61657-5_6.
- Jalilian, E., Uhl, A., & Kwitt, R. (2017). Domain adaptation for CNN based iris segmentation. In *2017 International conference of the biometrics special interest group* (pp. 1–6). <http://dx.doi.org/10.23919/BIOSIG.2017.8053502>.
- Jan, F. (2017). Segmentation and localization schemes for non-ideal iris biometric systems. *Signal Processing*. <http://dx.doi.org/10.1016/j.sigpro.2016.11.007>.
- Jillela, R., & Ross, A. A. (2013). Methods for iris segmentation. In M. J. Burge, & K. W. Bowyer (Eds.), *Handbook of iris recognition* (pp. 239–279). London: Springer London. http://dx.doi.org/10.1007/978-1-4471-4402-1_13.
- Kong, W. K., & Zhang, D. (2001). Accurate iris segmentation based on novel reflection and eyelash detection model. In *Proceedings of 2001 international symposium on intelligent multimedia, video and speech processing* (IEEE Cat. No.01EX489), (pp. 263–266). <http://dx.doi.org/10.1109/ISIMP.2001.925384>.
- Lemley, J., Bazrafkan, S., & Corcoran, P. (2017a). Deep learning for consumer devices and services: Pushing the limits for machine learning, artificial intelligence, and computer vision. *IEEE Consumer Electronics Magazine*, 6(2), 48–56. <http://dx.doi.org/10.1109/MCE.2016.2640698>.
- Lemley, J., Bazrafkan, S., & Corcoran, P. (2017b). Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*. <http://dx.doi.org/10.1109/ACCESS.2017.2696121>.
- Lemley, J., Bazrafkan, S., & Corcoran, P. (2017c). Transfer learning of temporal information for driver action classification. In *The 28th modern artificial intelligence and cognitive science conference*.
- Li, F., Zhang, B., & Liu, B. (2016). Ternary weight networks. arXiv Preprint arXiv:1605.04711.
- Liu, N., Li, H., Zhang, M., Liu, J., Sun, Z., & Tan, T. (2016). Accurate iris segmentation in non-cooperative environments using fully convolutional networks. In *2016 international conference on biometrics*, <http://dx.doi.org/10.1109/ICB.2016.7550055>.

- Liu, N., Zhang, M., Li, H., Sun, Z., & Tan, T. (2016). DeepIris: Learning pairwise filter bank for heterogeneous iris verification. *Pattern Recognition Letters*, 82, 154–161. <http://dx.doi.org/10.1016/j.patrec.2015.09.016>.
- Ma, L., Wang, Y., & Tan, T. (2002). Iris recognition using circular symmetric filters. *Object Recognition Supported By User Interaction for Service Robots*, 2, 414–417. <http://dx.doi.org/10.1109/ICPR.2002.1048327>.
- Matey, J. R., Naroditsky, O., Hanna, K., Kolczynski, R., Lolocono, D. J., Mangru, S., et al. (2006). Iris on the move: Acquisition of images for iris recognition in less constrained environments. *Proceedings of the IEEE*, 94. <http://dx.doi.org/10.1109/JPROC.2006.884091>.
- Menotti, D., Chiachia, G., Pinto, A., Schwartz, W. R., Pedrini, H., Falcão, A. X., et al. (2015). Deep representations for Iris, face, and fingerprint spoofing detection. *IEEE Transactions on Information Forensics and Security*, 10(4), 864–879. <http://dx.doi.org/10.1109/TIFS.2015.2398817>.
- Minaee, S., Abdolrashidiy, A., & Wang, Y. (2017). An experimental study of deep convolutional features for iris recognition. In *2016 IEEE signal processing in medicine and biology symposium, SPMB 2016 - proceedings* <http://dx.doi.org/10.1109/SPMB.2016.7846859>.
- MIRLIN. (n.d.). Retrieved from <https://www.fotonation.com/products/biometrics/iris-recognition/>.
- Morley, D., & Foroosh, H. (2017). Improving ransac-based segmentation through cnn encapsulation. In *Proc. IEEE conf. on computer vision and pattern recognition*.
- Pando, A. (2017). Beyond Security: Biometrics Integration Into Everyday Life. Retrieved November 21, 2017, from <https://www.forbes.com/sites/forbestechcouncil/2017/08/04/beyond-security-biometrics-integration-into-everyday-life/#ce400be431fb>.
- Phillips, C., & Komogortsev, O. V. (2011). Impact of Resolution and Blur on Iris Identification.
- Prabhakar, S., Pankanti, S., & Jain, A. K. (2003). Biometric recognition: security and privacy concerns. *IEEE Security & Privacy Magazine*, 1(2), 33–42. <http://dx.doi.org/10.1109/MSECP.2003.1193209>.
- Proenca, H. (2010). Iris recognition: On the segmentation of degraded images acquired in the visible wavelength. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8), 1502–1516.
- Proença, H., & Alexandre, L. A. (2006). Iris segmentation methodology for non-cooperative recognition. *IEE proceedings - vision, image, and signal processing*. <http://dx.doi.org/10.1049/ip-vis:20050213>.
- Proença, H., & Alexandre, L. A. (2010). Iris recognition: Analysis of the error rates regarding the accuracy of the segmentation stage. *Image and Vision Computing*, 28(1), 202–206. <http://dx.doi.org/10.1016/j.imavis.2009.03.003>.
- Proenca, H., Filipe, S., Santos, R., Oliveira, J., & Alexandre, L. A. (2010). The {UBIRIS.v2}: A database of visible wavelength images captured on-the-move and at-a-distance. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 32(8), 1529–1535. <http://dx.doi.org/10.1109/TPAMI.2009.66>.
- Quinn, G. W., Grother, P. J., Ngan, M. L., & Matey, J. R. (2013). IREX IV: part 1, evaluation of iris identification algorithms. NIST Interagency/Internal Report-7949.
- Radman, A., Zainal, N., & Suandi, S. A. (2017). Automated segmentation of iris images acquired in an unconstrained environment using HOG-SVM and GrowCut. *Digital Signal Processing*, 64, 60–70.
- Rakshit, S. (2007). *Novel methods for accurate human iris recognition*. University of Bath.
- Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, Part III* (pp. 234–241). Cham: Springer International Publishing. http://dx.doi.org/10.1007/978-3-319-24574-4_28.
- Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815–823).
- Sequeira, A. F., Monteiro, J. C., Rebelo, A., & Oliviera, H. P. (2014). MobBIO: A multi-modal database captured with a portable handheld device. In *9th international conference on computer vision theory and applications* (pp. 133–139). Lisbon.
- Shah, S., & Ross, a. (2009). Iris segmentation using geodesic active contours. *IEEE Transactions on Information Forensics and Security*, 4(4), 824–836. <http://dx.doi.org/10.1109/TIFS.2009.2033225>.
- Silva, P., Luz, E., Baeta, R., Pedrini, H., Falcao, A. X., & Menotti, D. (2015). An approach to iris contact lens detection based on deep image representations. In *Brazilian symposium of computer graphic and image processing, Vol. 2015–Octob* (pp. 157–164). <http://dx.doi.org/10.1109/SIBGRAPI.2015.16>.
- Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1), 1929–1958.
- Szegedy, C., Toshev, A., & Erhan, D. (2013). Deep neural networks for object detection. In *Advances in Neural Information Processing Systems* (pp. 2553–2561).
- Tan, T., He, Z., & Sun, Z. (2010). Efficient and robust segmentation of noisy iris images for non-cooperative iris recognition. *Image and Vision Computing*, 28(2), 223–230.
- Tan, C.-W., & Kumar, A. (2012). Unified framework for automated iris segmentation using distantly acquired face images. *IEEE Transactions on Image Processing*, 21(9), 4068–4079.
- Tan, C.-W., & Kumar, A. (2013). Towards online iris and periocular recognition under relaxed imaging constraints. *IEEE Transactions on Image Processing*, 22(10), 3751–3765.
- Thavalengal, S., Bigioi, P., & Corcoran, P. (2015a). Evaluation of combined visible/NIR camera for iris authentication on smartphones. In *2015 IEEE conference on computer vision and pattern recognition workshops*, (pp. 42–49). <http://dx.doi.org/10.1109/CVPRW.2015.7301318>.
- Thavalengal, S., Bigioi, P., & Corcoran, P. (2015b). Iris authentication in handheld devices - considerations for constraint-free acquisition. *IEEE Transactions on Consumer Electronics*, 61(2), 245–253. <http://dx.doi.org/10.1109/TCE.2015.7150600>.
- Thavalengal, S., & Corcoran, P. (2016). User authentication on smartphones: Focusing on iris biometrics. *IEEE Consumer Electronics Magazine*, 5(2), 87–93. <http://dx.doi.org/10.1109/MCE.2016.2522018>.
- Tisse, C.-, Martin, L., Torres, L., Robert, M., Zi, S., & Rousset, R. et al., (1992). Person identification technique using human iris recognition Advanced System Technology, (i).
- Wildes, R. P., Asmuth, J. C., Hanna, K. J., Hsu, S. C., Kolczynski, R. J., & Matey, J. R. et al., (1996). Automated, non-invasive iris recognition system and method. US Patent US5572596 A.
- Zhao, Z., & Ajay, K. (2015). An accurate iris segmentation framework under relaxed imaging constraints using total variation model. In *Proceedings of the IEEE international conference on computer vision* (pp. 3828–3836).
- Zheng, S., Song, Y., Leung, T., & Goodfellow, I. (2016). Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4480–4488).

Appendix E

Smart Augmentation Learning an Optimal Data Augmentation Strategy

Received February 16, 2017, accepted April 11, 2017, date of publication April 24, 2017, date of current version May 17, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2696121

Smart Augmentation Learning an Optimal Data Augmentation Strategy

JOSEPH LEMLEY, (Student Member, IEEE), SHABAB BAZRAFKAN, (Student Member, IEEE), AND PETER CORCORAN, (Fellow, IEEE)

Collage of Engineering and Informatics, National University of Ireland Galway, SW4 794 Galway, Ireland

Corresponding author: Joseph Lemley (j.lemley2@nuigalway.ie)

This work was supported by the SFI Strategic Partnership Program on Next Generation Imaging for Smartphone and Embedded Platforms by Science Foundation Ireland and FotoNation Ltd., under Project 13/SPP/I2868, and in part by the Irish Research Council Employment Based Programme Award under Project EBPPG/2016/280.

ABSTRACT A recurring problem faced when training neural networks is that there is typically not enough data to maximize the generalization capability of deep neural networks. There are many techniques to address this, including data augmentation, dropout, and transfer learning. In this paper, we introduce an additional method, which we call smart augmentation and we show how to use it to increase the accuracy and reduce over fitting on a target network. Smart augmentation works, by creating a network that learns how to generate augmented data during the training process of a target network in a way that reduces that networks loss. This allows us to learn augmentations that minimize the error of that network. Smart augmentation has shown the potential to increase accuracy by demonstrably significant measures on all data sets tested. In addition, it has shown potential to achieve similar or improved performance levels with significantly smaller network sizes in a number of tested cases.

INDEX TERMS Artificial intelligence, artificial neural networks, machine learning, computer vision supervised learning, machine learning algorithms, image databases.

I. INTRODUCTION

In order to train a deep neural network, the first and probably most important task is to have access to enough labeled samples of data. Not having enough quality labeled data will generate overfitting, which means that the network is highly biased to the data it has seen in the training set and, therefore will not be able to generalize the learned model to any other samples. In [1] there is a discussion about how much the diversity in training data and mixing different datasets can affect the model generalization. Mixing several datasets might be a good solution, but it is not always feasible due to lack of accessibility.

One of the other approaches to solving this problem is using different regularization techniques. In recent years different regularization approaches have been proposed and successfully tested on deep neural network models. The dropout technique [2] and batch normalization [3] are two well-known regularization methods used to avoid overfitting when training deep models.

Another technique for addressing this problem is called augmentation. Data augmentation is the process of supplementing a dataset with similar data that is created from the information in that dataset. The use of augmentation in deep learning is ubiquitous, and when dealing with images, often

includes the application of rotation, translation, blurring and other modifications to existing images that allow a network to better generalize [4].

Augmentation serves as a type of regularization, reducing the chance of overfitting by extracting more general information from the database and passing it to the network. One can classify the augmentation methods into two different types. The first is unsupervised augmentation. In this type of augmentation, the data expansion task is done regardless of the label of the sample. For example adding a different kind of noise, rotating or flipping the data. These kinds of data augmentations are usually not difficult to implement.

One of the most challenging kinds of data expansion is mixing different samples with the same label in feature space in order to generate a new sample with the same label. The generated sample has to be recognizable as a valid data sample, and also as a sample representative of that specific class. Since the label of the data is used to generate the new sample, this kind of augmentation this can be viewed as a type of supervised augmentation.

Many deep learning frameworks can generate augmented data. For example, Keras [5] has a built in method to randomly flip, rotate, and scale images during training but not all of these methods will improve performance and should not be

used “blindly”. For example, on MNIST (The famous handwritten number dataset), if one adds rotation, the network will be unable to distinguish properly between handwritten “6” and “9” digits. Likewise, a system that uses deep learning to classify or interpret road signs may become incapable of discerning left and right arrows if the training set was augmented with by indiscriminate flipping of images.

More sophisticated types of augmentation, such as selectively blending images or adding directional lighting rely on expert knowledge. Besides intuition and experience, there is no universal method that can determine if any specific augmentation strategy will improve results until after training. Since training deep neural nets is a time-consuming process, this means only a limited number of augmentation strategies will likely be attempted before deployment of a model.

Blending several samples in the dataset in order to highlight their mutual information is not a trivial task in practice. Which samples should be mixed together how many of them and how they mixed is a big problem in data augmentation using blending techniques.

Augmentation is typically performed by trial and error, and the types of augmentation performed are limited to the imagination, time, and experience of the researcher. Often, the choice of augmentation strategy can be more important than the type of network architecture used [6].

Before Convolutional Neural Networks (CNN) became the norm for computer vision research, features were “handcrafted”. Handcrafting features went out of style after it was shown that Convolutional Neural Networks could learn the best features for a given task. We suggest that since the CNN can generate the best features for some specific pattern recognition tasks, it might be able to give the best feature space in order to merge several samples in a specific class and generate a new sample with the same label. Our idea is to generate the merged data in a way that produces the best results for a specific target network through the intelligent blending of features between 2 or more samples.

II. RELATED WORK

Manual augmentation techniques such as rotating, flipping and adding different kinds of noise to the data samples, are described in depth in [4] and [7] which attempt to measure the performance gain given by specific augmentation techniques. They also provide a list of recommended data augmentation methods.

In 2014, Srivastava *et al.* introduced the dropout technique [2] aiming to reduce overfitting, especially in cases where there is not enough data. Dropout works by temporarily removing a unit (or artificial neuron) from the Artificial Neural Network and any connections to or from that unit.

Konda *et al.* Showed that dropout can be used for data augmentation by “projecting the the dropout noise within a network back into the input space” [8].

Jaderberg *et al.* devised an image blending strategy as part of their paper “Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition” [9]. They

used what they call “natural data blending” where each of the image layers is blended with a randomly sampled crop of an image from a training dataset. They note a significant (+44%) increase in accuracy using such synthetic images when image layers are blended together via a random process.

Another related technique is training on adversarial examples. Goodfellow *et al.* note that, although augmentation is usually done with the goal of creating images that are as similar as possible to the natural images one expects in the testing set, this does not need to be the case. They further demonstrate that training with adversarial examples can increase the generalization capacity of a network, helping to expose and overcome flaws in the decision function [10].

The use of Generative Adversarial Neural Networks [11] is a very powerful unsupervised learning technique that uses a min-max strategy wherein a ‘counterfeiter’ network attempts to generate images that look enough like images within a dataset to ‘fool’ a second network while the second network learns to detect counterfeits. This process continues until the synthetic data is nearly indistinguishable from what one would expect real data to look like. Generative Adversarial Neural Networks can also be used to generate images that augment datasets, as in the strategy employed by Shrivastav *et al.* [12].

Another method of increasing the generalization capacity of a neural network is called “transfer learning”. In transfer learning, we want to take knowledge learned from one network, and transfer it to another [13]. In the case of Convolutional Neural Networks, when used as a technique to reduce overfitting due to small datasets, it is common to use the trained weights from a large network that was trained for a specific task and to use it as a starting point for training the network to perform well on another task.

Batch normalization, introduced in 2015, is another powerful technique. It was discovered upon the realization that normalization need not just be performed on the input layer, but can also be achieved on intermediate layers [3].

Like the above regularization methods, Smart Augmentation attempts to address the issue of limited training data to improve regularization and reduce overfitting. As with [10], our method does not attempt to produce augmentations that appear “natural”. Instead, our network learns to combine images in ways that improve regularization. Unlike [4] and [7], we do not address manual augmentation, nor does our network attempt to learn simple transformations. Unlike the approach of image blending in [9], we do not arbitrarily or randomly blend images. Smart augmentation can be used in conjunction with other regularization techniques, including dropout and traditional augmentation.

III. SMART AUGMENTATION

Smart Augmentation is the process of learning suitable augmentations when training deep neural networks.

The goal of Smart Augmentation is to learn the best augmentation strategy for a given class of input data. It does this by learning to merge two or more samples

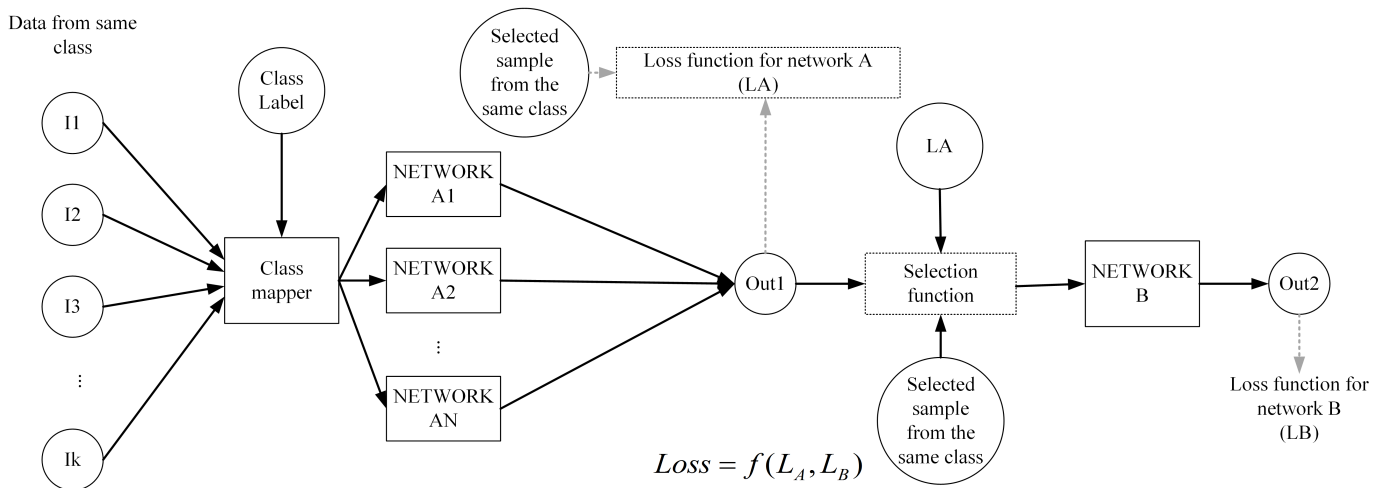


FIGURE 1. Smart augmentation with more than one network A.

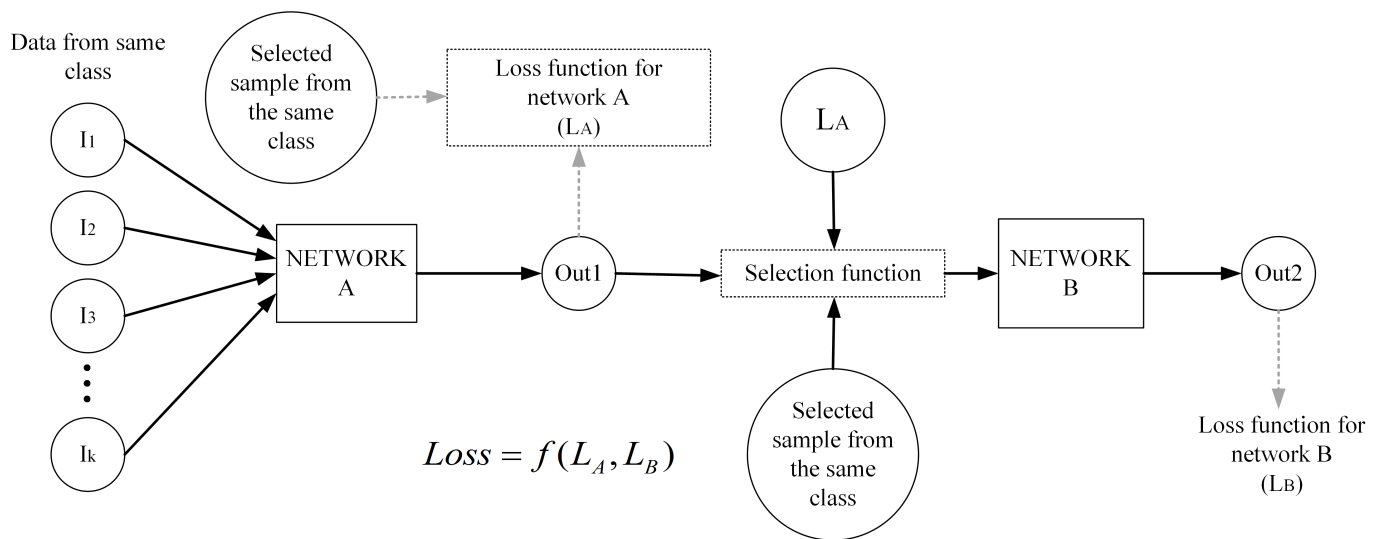


FIGURE 2. Diagram illustrating the reduced smart augmentation concept with just one network A.

in one class. This merged sample is then used to train a target network. The loss of the target network is used to inform the augmenter at the same time. This has the result of generating more data for use by the target network. This process often includes letting the network come up with unusual or unexpected but highly performant augmentation strategies.

A. TRAINING STRATEGY FOR SMART AUGMENTATION

During the training phase, we have two networks: Network A, which generates data; and network B, which is the network that will perform a desired task (such as classification). The main goal is to train network B to do some specific task while there are not enough representative samples in the given dataset. To do so, we use another network A to generate new samples.

This network accepts several inputs from the same class (the sample selection could be random, or it could use

some form of clustering, either in the pixel space or in the feature space) and generates an output which approximates data from that class. This is done by minimizing the loss function LA which accepts out1 and image i as input. Where out1 is the output of network A and image i is a selected sample from the same class as the input. The only constraint on the network A is that the input and output of this network should be the same shape and type. For example, if N samples of a P-channel image are fed to network A, the output will be a single P-channel image.

B. THE GENERATIVE NETWORK A AND LOSS FUNCTION

The loss function can be further parameterized by the inclusion of α and β as $f(L_A, L_B; \alpha, \beta)$. In the experiments and results sections of this paper, we examine how these can impact final accuracy.

Network A can either be implemented as a single network (figure 2) or as multiple networks, as in figure 1. Using

more than one network A has the advantage that the networks can learn class-specific augmentations that may not be suitable for other classes, but which work well for the given class.

Network A is a neural network, such as a generative model, with the difference that network A is being influenced by network B in the back propagation step, and network A accepts multiple samples as input simultaneously instead of just one at a time. This causes the data generated by network A to converge to the best choices to train network B for that specific task, and at the same time, it is controlled by loss function LA in a way that ensures that the outputs are similar to other members of its class.

The overall loss function during training is $f(LA, LB)$ where f is a function whose output is a transformation of LA and LB . This function could be an epoch-dependent function i.e. the function could change with the epoch number. In the training process, the error back-propagates from network B to network A. This tunes network A to generate the best augmentations for network B. After training is finished, Network A is cut out of the model and network B is used in the test process. The joint information between data samples is exploited to both reduce overfitting and to increase the accuracy of the target network during training.

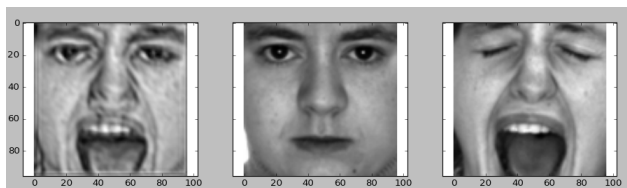


FIGURE 3. The image on the left is created by a learned combination of the two images on the right. This type of image transformation helped increase the accuracy of network B. The image was not produced to be an ideal approximation of a face but instead, contains features that helped network B better generalize the concept of gender which is the task it was trained for.

C. HOW SMART AUGMENTATION WORKS

The proposed method uses a network (network A) to learn the best sample blending for the specific problem. The output of network A is the used for the input of network B. The idea is to use network A to learn the best data augmentation to train network B. Network A accepts several samples from the same class in the dataset and generates a new sample from that class, and this new sample should reduce the training loss for network B. In figure 3 we see an output of network A designed to do the gender classification. The image on the left is a merged image of the other two. This image represents a sample from the class “male” that does not appear in the dataset, but still, has the identifying features of its class.

Notice that in figure 3, an image was created with an open mouth and open eyes from two images. The quality of the face image produced by network A does not matter. Only its ability to help network B better generalize. Our approach is most applicable to classification tasks but may also have applications in any approach where the selective blending of sample

features improves performance. Our observations show that this approach can reduce overfitting and increase accuracy. In the following sections, we evaluate several implementations of our smart augmentation technique on various datasets to show how it can improve accuracy and prevent overfitting. We also show that with smart augmentation, we can train a very small network to perform as well as (or better than) a much larger network that produces state of the art results.

IV. METHODS

Experiments were conducted on NVIDIA Titan X GPU’s running a pascal architecture with python 2.7, using the Theano [14] and Lasagne frameworks.

A. DATA PREPARATION

To evaluate our method, we chose 4 datasets with characteristics that would allow us to examine the performance of the algorithm on specific types of data. Since the goal of our paper is to measure the impact of the proposed technique, we do not attempt to provide a comparison of techniques that work well on these databases. For such a comparison we refer to [15] for gender datasets or [16] for the places dataset.

1) HIGHLY CONSTRAINED FACES DATASET (db1)

Our first dataset, db1 was composed from the AR faces database [17] with a total of 4,000 frontal faces of male and female subjects. The data was split into subject exclusive training, validation, and testing sets, with 70% for training, 20% for validation, and 10% for testing. All face images were reduced to 96X96 grayscale with pixel values normalized between 0 and 1.

2) AUGMENTED, HIGHLY CONSTRAINED FACES DATASET (db1a)

To compare traditional augmentation with smart augmentation and to examine the effect of traditional augmentation on smart augmentation, we created an augmented version of db1 with every combination of flipping, blurring, and rotation (-5,-2,0,2,5 degrees with the axis of rotation at the center of the image). This resulted in a larger training set of 48360 images. The test and validation sets were unaltered from db1. The data was split into a subject exclusive training, validation, and testing sets with 70% for training, 20% for validation, and 10% for testing. All face images were reduced to 96X96 with pixel values normalized between 0 and 1.

3) FERET

Our second dataset, db2, was the FERET dataset. We converted FERET to grayscale and reduced the size of each image to 100X100 with pixel values normalized between 0 and 1. The data was split into subject exclusive training, validation, and testing sets, with 70% for training, 20% for validation and 10% for testing.

Color FERET [18] Version 2 was collected between December 1993 and August 1996 and made freely available with the intent of promoting the development of face



FIGURE 4. Arbitrarily selected images from FERET demonstrate similarities in lighting, pose, subject, background, and other photographic conditions.

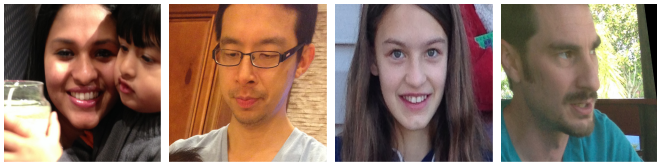


FIGURE 5. Arbitrarily selected images from the Adience show significant variations in lighting, pose, subject, background, and other photographic conditions.

recognition algorithms. The images are labeled with gender, pose, and name.

Although FERET contains a large number of high-quality images in different poses and with varying face obstructions (beards, glasses, etc), they all have certain similarities in quality, background, pose, and lighting that make them very easy for modern machine learning methods to correctly classify (see figure 4). In our experiments, we use all images in FERET for which gender labels exist.

4) ADIENCE

Our third dataset, db3, was Adience (see figure 5). We converted Adience to grayscale images with size 100×100 and normalized the pixel values between 0 and 1. The data was split into subject exclusive training, validation, and testing sets, with 70% for training, 20% for validation and 10% for testing.

5) DB4

Our fourth dataset, db4, was the MIT places dataset [16]. The MIT PLACES dataset is a machine learning database containing has 205 scene categories and 2.5 million labeled images.

The Places Dataset is unconstrained and includes complex scenery in a variety of lighting conditions and environments, as shown in figure 6.

We restricted ourselves to just the first two classes in the dataset (Abbey and Airport). Pixel values were normalized between 0 and 1. The “small dataset,” which had been rescaled to 256×256 with 3 color channels, was used for all experiments without modification except for normalization of the pixel values between 0 and 1.

V. EXPERIMENTS

In these experiments, we call network B the network that is being trained for a specific task (such as classification).



FIGURE 6. Example images from the MIT places dataset showing two examples from each of the two classes (abbey and airport) used in our experiments.

TABLE 1. Full listing of experiments.

Exp	DB	Net A	A ch	Net B	α	β	LR	Momentum
1	db1	1	1	B_1	0.3	0.7	0.01	0.9
2	db1	1	2	B_1	0.3	0.7	0.01	0.9
3	db1	1	3	B_1	0.3	0.7	0.01	0.9
4	db1	1	4	B_1	0.3	0.7	0.01	0.9
5	db1	1	5	B_1	0.3	0.7	0.01	0.9
6	db1	1	6	B_1	0.3	0.7	0.01	0.9
7	db1	1	7	B_1	0.3	0.7	0.01	0.9
8	db1	1	8	B_1	0.3	0.7	0.01	0.9
9	db1	2	1	B_1	0.3	0.7	0.005	0.9
10	db1	2	2	B_1	0.3	0.7	0.005	0.9
11	db1	2	3	B_1	0.3	0.7	0.005	0.9
12	db1	2	4	B_1	0.3	0.7	0.005	0.9
13	db1	2	5	B_1	0.3	0.7	0.005	0.9
14	db1	2	6	B_1	0.3	0.7	0.005	0.9
15	db1	2	7	B_1	0.3	0.7	0.005	0.9
16	db1	2	8	B_1	0.3	0.7	0.005	0.9
17	db1	NA	NA	B_1	NA	NA	0.01	0.9
18	db1a	NA	NA	B_1	NA	NA	0.01	0.9
19	db1a	1	2	B_1	0.3	0.7	0.01	0.9
20	db1a	2	2	B_1	0.3	0.7	0.005	0.9
21	db2	NA	NA	B_1	NA	NA	0.01	0.9
22	db2	1	2	B_1	0.3	0.7	0.01	0.9
23	db3	NA	NA	B_1	NA	NA	0.01	0.9
24	db3	1	2	B_1	0.3	0.7	0.01	0.9
25	db4	NA	NA	B_2	NA	NA	0.005	0.9
26	db4	NA	NA	B_1	NA	NA	0.005	0.9
27	db4	1	2	B_1	0.3	0.7	0.01	0.9
28	db4	1	2	B_1	0.7	0.3	0.01	0.9
29	db4	2	2	B_1	0.7	0.3	0.005	0.9
30	db4	2	2	B_1	0.3	0.7	0.005	0.9

We call network A the network that learns augmentations that help train network B.

All experiments are run for 1000 epochs. The test accuracy reported is for the network that had the highest score on the validation set during those 1000 epochs.

To analyze the effectiveness of Smart Augmentation, we performed 30 experiments using 4 datasets with different parameters. A brief overview of the experiments can be seen in Table I. The experiments were conducted with the motivation of answering the following questions:

- 1) Is there any difference in accuracy between using smart augmentation and not using it? (Is smart augmentation effective?)
- 2) If smart augmentation is effective, is it effective on a variety of datasets?
- 3) As the datasets become increasingly unconstrained, does smart augmentation perform better or worse?
- 4) What is the effect of increasing the number of channels in the smart augmentation method?
- 5) Can smart augmentation improve accuracy over traditional augmentation?
- 6) If smart augmentation and traditional augmentation are combined, are the results better or worse than not combining them?

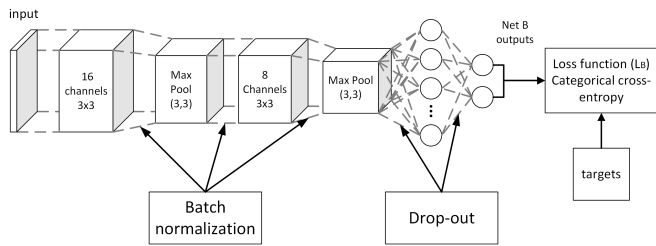


FIGURE 7. Illustration of network B₁.

- 7) Does altering the α and β parameters change the results?
- 8) Does Smart Augmentation increase or decrease overfitting as measured by train/test loss ratios?
- 9) If smart augmentation decreases overfitting, can we use it to replace a large complex network with a simpler one without losing accuracy?
- 10) What is the effect of the number of network A's on the accuracy? Does training separate networks for each class improve the results?

As listed below, we used three neural network architectures with varied parameters and connection mechanisms. In our experiments, these architectures were combined in various ways as specified in table 1.

- Network B₁ is a simple, small Convolutional neural network, trained as a classifier, that takes an image as input, and outputs class labels with a softmax layer. This network is illustrated in figure 7.
- Network B₂ is a unmodified implementation of VGG16 as described in [19]. Network B₂ is a large network that takes an image as input and outputs class labels with a softmax layer.
- Network A is a Convolutional neural network that takes one or more images as input and outputs a modified image.

A. SMART AUGMENTATION WITH ONE NETWORK A ON THE GENDER CLASSIFICATION TASK

Experiments 1-8, 19,22, and 24 as seen in table 1 were trained for gender classification using the same technique as illustrated in figure 9. In these experiments, we use smart augmentation to train a network (network B) for gender classification using the specified database.

The first, k images are randomly selected from the same class (male or female) in the dataset. These k samples are merged into k channels of a single sample. The grayscale values of the first image, img_0 , are mapped to channel 0 and the grayscale values of the second image im_1 are mapped to channel 1 and so on until we reach the number of channels specified in the experiments table. This new k channel image is fed into the network A. Network A is a fully convolutional neural network (See figure 8) which accepts images as the input and gives the images with the same size at the output in a single channel.

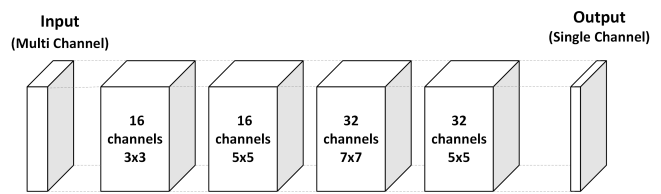


FIGURE 8. Illustration of network A.

An additional grayscale image is then randomly selected from the same class in the dataset (this image should not be any of those images selected in step 1). The loss function for this network A is calculated as the mean squared error between this randomly selected image and the output of network A. The output of network A, and the target image is then fed into network B as separate inputs. Network B is a typical deep neural network with two convolutional layers followed by batch normalization and max-pooling steps after each convolutional layer. Two fully connected layers are placed at the end of the network. The first of these layers has 1024 units and the second dense layer is made of two units as the output of network B using softmax. Each dense layer takes advantage of the drop-out technique in order to avoid overfitting. The loss function of network B is calculated as the categorical cross-entropy between the outputs and the targets.

The total loss of the whole model is a linear combination of the loss functions of two networks. This approach is designed to train a network A that generates samples which reduce the error for network B. The validation loss was calculated only for network B, without considering network A. This allows us to compare validation loss with and without smart augmentation.

Our models were trained using Stochastic Gradient Descent with Nesterov Momentum [20], learning rate 0.01 and momentum 0.9. The lasagne library used to train the network in python.

In these experiments, we varied the number of input channels and datasets used. Specifically, we trained a network B from scratch with 1-8 input channels with a single network A on db1, 2 channels on network A for db2 and 3, and 2 channels on network db1a as shown in the table of experiments.

B. SMART AUGMENTATION WITH TWO NETWORK A'S ON THE GENDER CLASSIFICATION TASK

In experiments 9-16 and 20 we evaluate a different implementation of smart augmentation, containing a separate network A for each class. As before, the first k images are randomly selected from the same class (male or female) in the dataset. These k samples are merged into k channels of a single sample. The grayscale values of the first image, img_0 , are mapped to channel 0 and the grayscale values of the second image, im_1 , are mapped to channel 1, and so on until we reach the number of channels specified in the experiments table just as before. Since we now have two network A's, it is important to separate out the loss functions for each network as illustrated in figure 10.

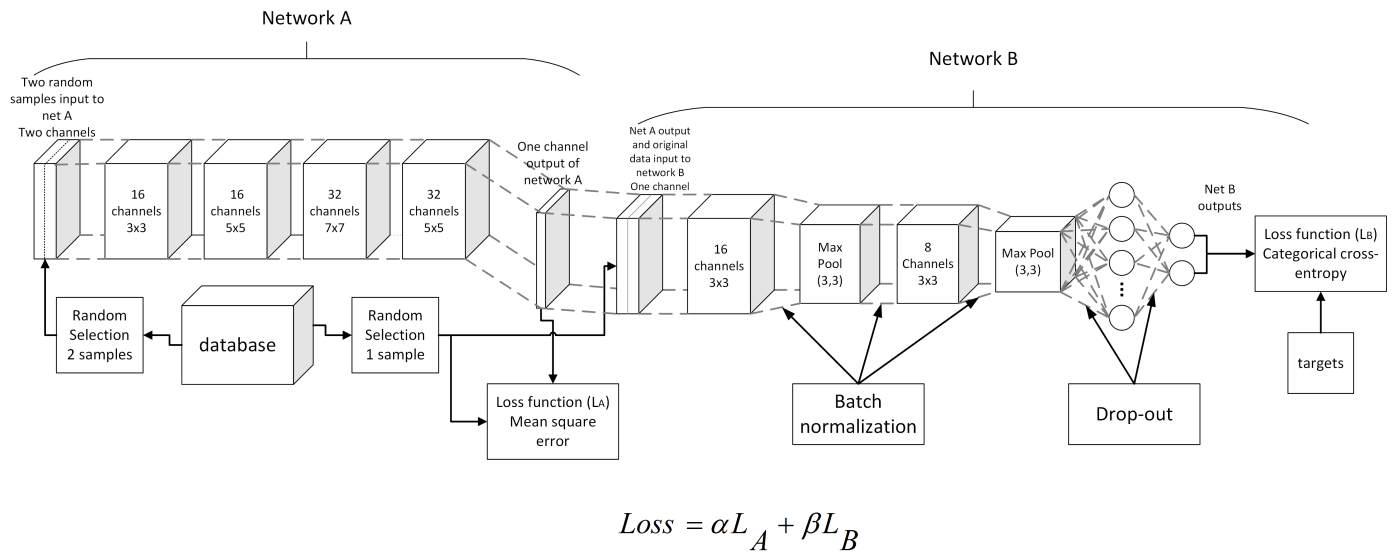


FIGURE 9. Diagram of simplified implementation of Smart Augmentation showing network A and network B.

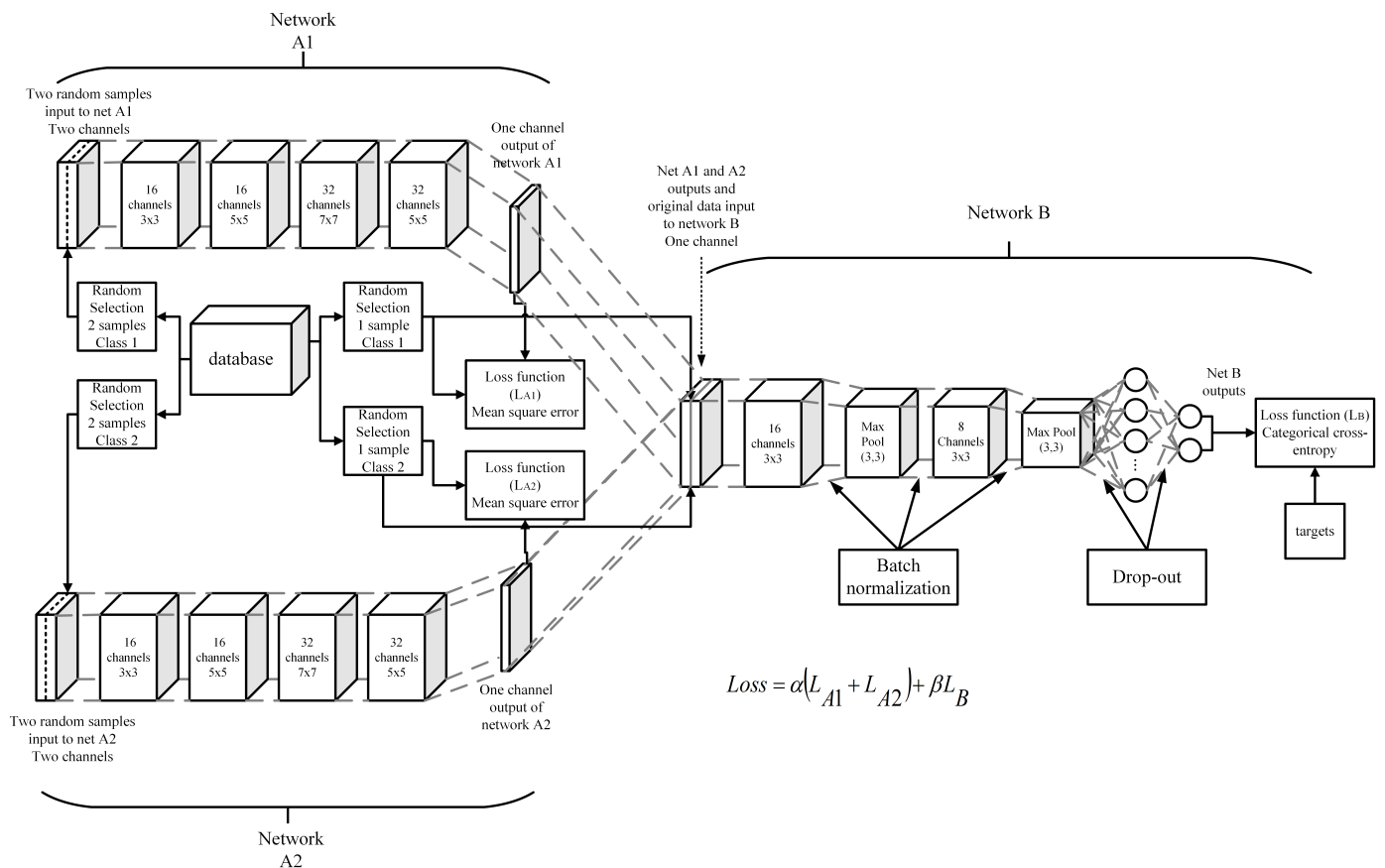


FIGURE 10. Diagram of our implementation of Smart Augmentation with one network A for each class.

All other loss functions are calculated the same way as before.

One very important difference is the updated learning rate (0.005). While performing initial experiments we noticed that using a learning rate above 0.005 led to the “dying

RELU” problem and stopped effective learning within the first two epochs. This network is also more sensitive to variations in batch size.

The goal of these experiments was to examine how using multiple network As impacts accuracy and overfitting

compared to just using one network A. We also wanted to know if there were any differences when trained on a manually augmented database (experiment 20).

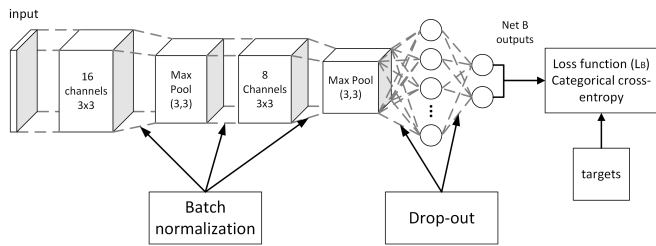


FIGURE 11. Diagram of implementation of network B without Smart Augmentation.

C. TRAINING WITHOUT SMART AUGMENTATION ON THE GENDER CLASSIFICATION TASK

In these experiments, we train a network [network B (see figure 11)] to perform gender classification without applying network A during the training stage. These experiments (23, 21, 18, and 17) are intended to serve as a baseline comparison of what network B can learn without smart augmentation on a specific dataset (db3,db2, db1a, and db1 respectively). In this way, we measure any improvement given by smart augmentation. A full implementation of Network B is shown in figure 7.

This network has the same architecture as the network B presented in the previous experiment except that it does not utilize a network A.

As before, two fully connected layers are placed at the end of the network. The first of these layers has 1024 units, and the second dense layer has two units (one for each class). Each dense layer takes advantage of the drop-out technique in order to avoid over-fitting.

All loss functions (training, validation, and testing loss) were calculated as the categorical cross-entropy between the outputs and the targets.

As before, models were trained using Stochastic Gradient Descent with Nesterov Momentum [20], learning rate 0.01 and momentum 0.9. The lasagne library was used to train the network in python.

D. EXPERIMENTS ON THE PLACES DATASET

In the previous experiments in this section, we used 3 different face datasets. In experiments 25 - 30 we examine the suitability of Smart Augmentation with color scenes from around the world from the MIT Places dataset to evaluate our method on data of a completely different topic. We varied the α and β parameter in our global loss function so that we could identify how they influence results. Unlike in previous experiments, we also retained color information.

Experiment 25 utilized a VGG16 trained from scratch as a classifier, chosen because VGG16 models have performed very well on the places dataset in public competitions [16]. The input to network A was 256×256 RGB images and the output was determined by a 2 class softmax classifier.

In experiment 26 we use a network B, identical in all respects to the one used in the previous subsection, except that we use the lower learning rate specified in the experiments table and take in color images about places instead of gender.

These two experiments (25,26) involved simple classifiers to establish a baseline against which other experiments on the same dataset could be evaluated.

In experiments 27-28, k images were randomly selected from the same class (abbey or airport) in the dataset. These k samples are merged into $k \times 3$ channels of a single sample. The values of the first three channels of image img_0 are mapped to channel 0-2, and the first three channels of the second image im_1 are mapped to channels 3-5, and so on, until we reach the number of channels specified in the experiments table multiplied by the number of color channels in the source images. This new $k \times 3$ channel image is used by network A. Network A is a fully convolutional neural network which accepts images as the input, and outputs just one image.

An additional image is then randomly selected from the same class in the dataset. The loss function for network A is calculated as the mean squared error between the randomly selected image and the output of network A. The output of network A, and the target image is then fed into network B as separate inputs. Network B is a typical deep neural network with two convolutional layers followed by batch normalization and max-pooling steps after each convolutional layer. Two fully connected layers are placed at the end of the network. The first of these layers has 1024 units and the second dense layer is made of two units as the output of network B using softmax. Each dense layer takes advantage of the drop-out technique in order to avoid over-fitting. The loss function of network B is calculated as the categorical cross-entropy between the outputs and the targets.

The total loss of the whole model is a linear combination of the loss functions of two networks. This approach is designed to train a network A that generates samples that reduce the error for network B. The validation loss was calculated only for network B, without considering network A. This allows us to compare validation loss with and without smart augmentation.

Our models were trained using Stochastic Gradient Descent with Nesterov Momentum [20], learning rate 0.005 and momentum 0.9. The Lasagne library was used to train the network in python.

In these experiments, we varied the number of input channels and datasets used. Specifically, we trained a network B from scratch with 1-8 input channels on network A on db1, 2 channels on network A for db2 and 3, and 2 channels on network db1a as shown in the table of experiments.

In experiments 29-30, k images are randomly selected from the same class (abbey or airport) in the dataset. These k samples are merged into $k \times 3$ channels of a single sample. The values of the first three channels in image img_0 are mapped to channel 0-2 and the first three channels of the second image im_1 are mapped to channels 3-5 and so

on until we reach the number of channels specified in the experiments table multiplied by the number of color channels in the source images. This new $k \times 3$ channel image is fed into the network A. Network A is a fully convolutional neural network which accepts images as the input and outputs a single color image.

An additional image is then randomly selected from the same class in the dataset. The loss function for each network A is calculated as the mean squared error between the randomly selected image and the output of network A. The output of network A, and the target image is then fed into network B as separate inputs. Network B is a typical deep neural network with two convolutional layers followed by batch normalization and max-pooling steps after each convolutional layer. Two fully connected layers are placed at the end of the network. The first of these layers has 1024 units, and the second dense layer is made of two units as the output of network B using softmax. Each dense layer takes advantage of the drop-out technique in order to avoid over-fitting. The loss function of network B is calculated as the categorical cross-entropy between the outputs and the targets.

The total loss of the whole model is a linear combination of the loss functions of the two networks. This approach is designed to train a network A that generates samples that reduce the error for network B. The validation loss was calculated only for network B, without considering network A. This allows us to compare validation loss with and without smart augmentation.

Our models were trained using Stochastic Gradient Descent with Nesterov Momentum [20], learning rate 0.005 and momentum 0.9. The lasagne library was used to train the network in python.

In these experiments, we varied the number of input channels and datasets used. Specifically, we trained a network B from scratch with 1-8 input channels on network A on db1, 2 channels on network A for db2 and 3, and 2 channels on network db1a as shown in the table of experiments.

VI. RESULTS

The results of experiments 1-30 as shown in Table 1 are listed in tables 2 and 3 and are listed in the same order as in the corresponding experiments table. These results are explained in detail in the subsections below.

A. SMART AUGMENTATION WITH ONE NETWORK A ON THE GENDER CLASSIFICATION TASK

In figure 12, we show the training and validation loss for experiments 1 and 17. As can be observed, the rate of overfitting was greatly reduced when smart augmentation was used compared to when it was not used.

Without smart augmentation, network B had an accuracy of 88.15 for the AR faces dataset; for the rest of this subsection, this result is used as a baseline by which other results on that dataset are evaluated.

One can see how the smart augmentation technique could prevent network B from overfitting in the training stage.

TABLE 2. Results of experiments on face datasets.

Experiments on Face Datasets				
Dataset	#Net As	Input Channels	Augmented	Test Accuracy
AR Faces	1	1	no	0.927746
AR Faces	1	2	no	0.924855
AR Faces	1	3	no	0.950867
AR Faces	1	4	no	0.916185
AR Faces	1	5	no	0.910405
AR Faces	1	6	no	0.933526
AR Faces	1	7	no	0.916185
AR Faces	1	8	no	0.953757
AR Faces	2	1	no	0.869942188
AR Faces	2	2	no	0.956647396
AR Faces	2	3	no	0.942196548
AR Faces	2	4	no	0.942196548
AR Faces	2	5	no	0.907514453
AR Faces	2	6	no	0.933526039
AR Faces	2	7	no	0.916184962
AR Faces	2	8	no	0.924855471
AR Faces	0	NA	no	0.881502867
AR Faces	0	NA	yes	0.890173435
AR Faces	1	2	yes	0.956647396
AR Faces	2	2	yes	0.956647396
Adience	0	NA	no	0.700206399
Adience	1	2	no	0.760577917
FERET	0	NA	no	0.835242271
FERET	1	2	no	0.884581506

TABLE 3. Results of experiments on place dataset.

Experiments on MIT places dataset				
#Net As	Target Network	Test Accuracy	A	B
0	VGG16	98.5	NA	NA
0	Small net B	96.5	NA	NA
1	Small net B	98.75	0.3	0.7
1	Small net B	99%	0.7	0.3
2	Small net B	99%	0.7	0.3
2	Small net B	97.87%	0.3	0.7

The smaller difference between training loss and validation loss caused by the smart augmentation technique shows how this approach helps the network B to learn more general features for this task. Network B also had higher accuracy on the test set when trained with smart augmentation.

In figures 13 and 14 we show examples of the kinds of images network A learned to generate. In these figures, the image on the left side is the blended image of the other two images produced by network A.

We observe an improvement in accuracy from 83.52% to 88.46% from smart augmentation on Feret with 2 inputs and an increase from 70.02% to 76.06% on the adience dataset.

We see that there is no noticeable pattern when we vary the number of inputs for network A. Despite the lack of a pattern, a significant difference was observed with 8 and 3 channels providing the best results at 95.38% and 95.09% respectively. At the lower end, 7, 5, and 4 channels performed the worst, with accuracies of 91.62%, 91.04%, and 91.04%.

Recall that the accuracy without network A was: 88.15% for the AR faces dataset. We suspect that much of the variation in accuracy reported above may be due to chance. Since in this particular experiment, images are chosen randomly there may be times when 2 or more images with very helpful mutual information are present by chance and the opposite is also possible. It is interesting that when 3 and 8 channels were used for network A, the accuracy was over 95%.

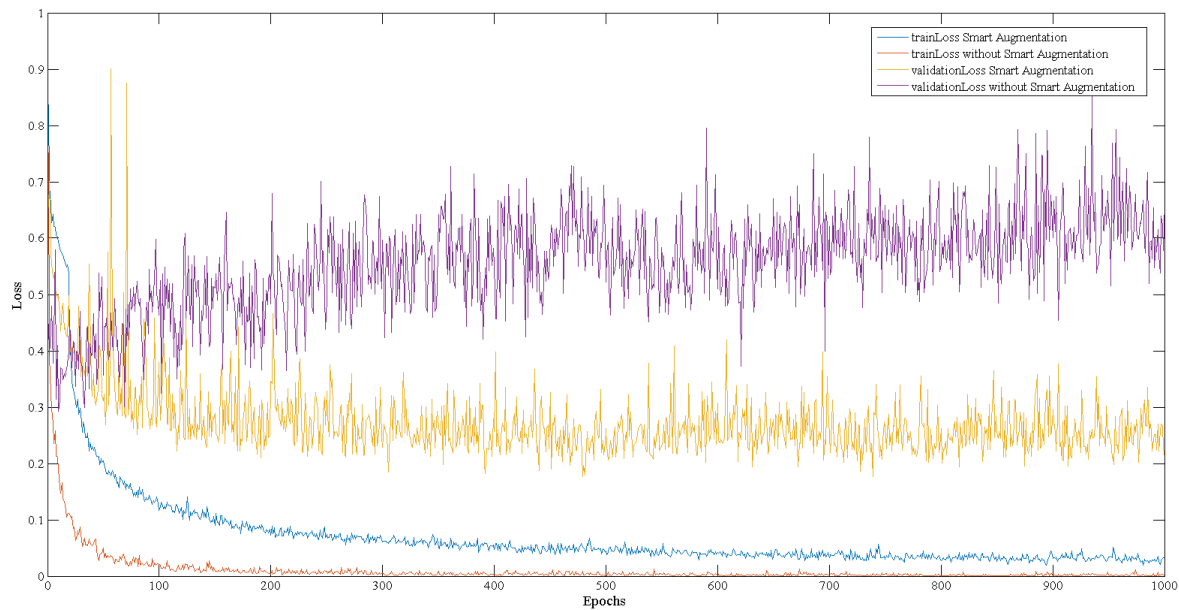


FIGURE 12. Training and validation losses for experiments 1 and 17, showing reductions in overfitting by using Smart Augmentation. The smaller difference between training loss and validation loss caused by the smart augmentation technique shows how this approach helps the network B to learn more general features for this task. To avoid confusion, we remind the reader that the loss for smart augmentation is given by $f(L_A, L_B; \alpha, \beta)$. This means that the loss graphs are a combination of the losses of two networks whereas the losses without smart augmentation are only $f(L_B)$.

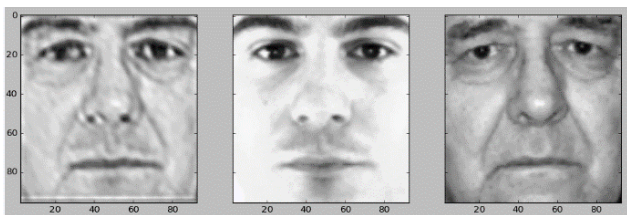


FIGURE 13. The image on the left is a learned combination of the two images on the right as produced by network A.



FIGURE 14. The image on the left is a learned combination of the two images on the right as produced by network A.

B. SMART AUGMENTATION AND TRADITIONAL AUGMENTATION

We note that traditional augmentation improved the accuracy from 88.15% to 89.08% without smart augmentation on the gender classification task. When we add smart augmentation we realize an improvement in accuracy to 95.66%.

The accuracy of the same experiment when we used 2 networks A’s was also 95.66% which seems to indicate that both configurations may have found the same optima when smart augmentation was combined with traditional augmentation.

This demonstrates that smart augmentation can be used with traditional augmentation to further improve accuracy. In all cases examined so far, Smart Augmentation performed better than traditional augmentation. However, since there are no practical limits on the types of traditional augmentation that can be performed, there is no way to guarantee that manual augmentation could not find a better augmentation strategy. This is not a major concern since we do not claim that smart augmentation should replace traditional augmentation. We only claim that smart augmentation can help with regularization.

C. SMART AUGMENTATION WITH TWO NETWORK A’S ON THE GENDER CLASSIFICATION TASK

In this subsection, we discuss the results of our two network architecture when trained on the gender classification set.

These experiments show that approaches which use a distinct network A for each class, tend to slightly outperform networks with just 1 network A. This seems to provide support for our initial idea that one network A should be used for each class so that class-specific augmentations could be more efficiently learned. If the networks with just 1 and 0 input channels are excluded, we see an average increase in accuracy from 92.94% to 93.19% when smart augmentation is used, with the median accuracy going from 92.49% to 93.35%.

There is only one experiment where smart augmentation performed worse than not using smart augmentation. This can be seen in the 9th row of table II where we use only one channel which caused the accuracy to dip to 86.99%, contrasted with 88.15% when no smart augmentation is used. This is expected because when only one channel is used, mutual

information can not be effectively utilized. This experiment shows the importance of always using at least 2 channels.

D. EXPERIMENTS ON THE PLACES DATASET

As with previously discussed results, when the places dataset is used, networks with multiple network A's performed slightly better. We also notice that when α is higher than β an increase in accuracy is realized.

The most significant results of this set of experiments is the comparison between smart augmentation, VGG 16, and network B trained alone. Note that a small network B trained alone (no Smart Augmentation) had an accuracy of 96.5% compared to VGG 16 (no Smart Augmentation) at 98.5%. When the same small network B was trained with smart augmentation we see accuracies ranging from 98.75% to 99% which indicates that smart augmentation, in some cases, can allow a much smaller network to replace a larger network.

VII. DISCUSSION AND CONCLUSION

Smart Augmentation has shown the potential to increase accuracy by demonstrably significant measures on all datasets tested. In addition, it has shown potential to achieve similar or improved performance levels with significantly smaller network sizes in a number of tested cases.

In this paper, we discussed a new regularization approach, called "Smart Augmentation" to automatically learn suitable augmentations during the process of training a deep neural network. We focus on learning augmentations that take advantage of the mutual information within a class. The proposed solution was tested on progressively more difficult datasets starting with a highly constrained face database and ending with a highly complex and unconstrained database of places. The various experiments presented in this work demonstrate that our method is appropriate for a wide range of tasks and demonstrates that it is not biased to any particular type of image data.

As a primary conclusion, these experiments demonstrate that the augmentation process can be automated, specifically in nontrivial cases where two or more samples of a certain class are merged in nonlinear ways resulting in improved generalization of a target network. The results indicate that a deep neural network can be used to learn the augmentation task in this way at the same time the task is being learned. We have demonstrated that smart augmentation can be used to reduce overfitting during the training process and reduce the error during testing.

It is worthwhile to summarize a number of additional observations and conclusions from the various experiments documented in this research.

Firstly, no linear correlation between the number of samples mixed by network A and accuracy was found so long as at least 2 samples are used.

Secondly, it was shown that Smart Augmentation is effective at reducing error and decreasing overfitting and that this is true regardless of how unconstrained the database is.

Thirdly, these experiments demonstrated that better accuracy could be achieved with smart augmentation than with traditional augmentation alone. It was found that altering the α and β parameters of the loss function slightly impacts results but more experiments are needed to identify if optimal parameters can be found.

Finally, it was found that Smart Augmentation on a small network achieved better results than those obtained by a much larger network (VGG 16). This will help enable more practical implementations of CNN networks for use in embedded systems and consumer devices where the large size of these networks can limit their usefulness.

Future work may include expanding Smart Augmentation to learn more sophisticated augmentation strategies and performing experiments on larger datasets with larger numbers of data classes. A statistical study to identify the number of channels that give the highest probability of obtaining optimal results could also be useful.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU used for this research.

REFERENCES

- [1] S. Bazrafkan, T. Nedelcu, P. Filipczuk, and P. Corcoran, "Deep learning for facial expression recognition: A step closer to a smartphone that knows your moods," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2017, pp. 217–220.
- [2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [3] S. Ioffe and C. Szegedy. (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." [Online]. Available: <https://arxiv.org/abs/1502.03167>
- [4] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. ICDAR*, vol. 3, Aug. 2003, pp. 958–962.
- [5] F. Chollet. (2015). *Keras*. [Online]. Available: <https://github.com/fchollet/keras>
- [6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: <http://www.deeplearningbook.org>.
- [7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. (2014). "Return of the devil in the details: Delving deep into convolutional nets." [Online]. Available: <http://arxiv.org/abs/1405.3531>
- [8] X. Bouthillier, K. Konda, P. Vincent, and R. Memisevic. (2015). "Dropout as data augmentation." [Online]. Available: <https://arxiv.org/abs/1506.08700>
- [9] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. (2014). "Synthetic data and artificial neural networks for natural scene text recognition." [Online]. Available: <http://arxiv.org/abs/1406.2227>
- [10] I. J. Goodfellow, J. Shlens, and C. Szegedy. (2014). "Explaining and harnessing adversarial examples." [Online]. Available: <https://arxiv.org/abs/1412.6572>
- [11] I. Goodfellow et al., "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
- [12] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. (2016). "Learning from simulated and unsupervised images through adversarial training." [Online]. Available: <https://arxiv.org/abs/1612.07828>
- [13] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[14] Theano Development Team. (May 2016). "Theano: A python framework for fast computation of mathematical expressions." [Online]. Available: <http://arxiv.org/abs/1605.02688>

[15] J. Lemley, S. Abdul-Wahid, D. Banik, and R. Andonie, "Comparison of recent machine learning techniques for gender recognition from facial images," in *Proc. 27th Modern Artif. Intell. Cognit. Sci. Conf.*, 2016, pp. 97–102.

[16] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 487–495.

[17] A. Martinez and R. Benavente, "The AR face database," CVC, Tech. Rep. #24, 1998. [Online]. Available: <http://www2.ece.ohio-state.edu/~aleix/ARdatabase.html>

[18] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET evaluation methodology for face-recognition algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 10, pp. 1090–1104, Oct. 2000.

[19] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: <https://arxiv.org/abs/1409.1556>

[20] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. ICML*, Feb. 2013, vol. 28, no. 3, pp. 1139–1147.



JOSEPH LEMLEY (S'17) received the B.S. degree in computer science and the master's degree in computational science from Central Washington University in 2006 and 2016, respectively. He is currently pursuing the Ph.D. degree with the National University of Ireland Galway. He was with the industry and co-founded a start-up. His field of work is machine learning using deep neural networks for tasks related to computer vision. His Ph.D. was funded by FotoNation, Ltd., under the IRCSET Employment Ph.D. Program.



SHABAB BAZRAFKAN (S'16) received the B.Sc. degree in electrical engineering from Urmia University, Urmia, Iran, in 2011, and the M.Sc. degree in telecommunication engineering, image processing branch from the Shiraz University of Technology in 2013. He is currently pursuing the Ph.D. degree with the National University of Ireland Galway. He is currently with FotoNation, Ltd. His field of working is deep neural networks and neural network design.



PETER CORCORAN (M'95–F'10) was the former Vice-Dean of Research and Graduate Studies (seven years tenure) with the College of Engineering and Informatics, National University of Ireland Galway. He has been a University Professor for 28 years. He is the co-founder of several start-up companies including FotoNation, industry consultant and expert witness. He has over 500 technical publications and patents, over 80 peer reviewed papers and articles, over 100 international conference papers. He has co-invented over 300 granted U.S. patents. He is a member of the IEEE Consumer Electronics Society for over 20 years. He is the Editor-in-Chief and the Founding Editor of the *IEEE Consumer Electronics Magazine*.

•••

Appendix F

Latent Space Mapping for Generation of Object Elements with Corresponding Data Annotation



Latent Space Mapping for Generation of Object Elements with Corresponding Data Annotation

Shabab Bazrafkan^{a*}, Hossein Javidnia^a, and Peter Corcoran^a

^a*Department of Electronic Engineering, College of Engineering, National University of Ireland Galway, University Road, Galway, Ireland*

ABSTRACT

Deep neural generative models such as Variational Auto-Encoders (VAE) and Generative Adversarial Networks (GAN) are giving promising results in estimating the data distribution across a range of machine learning fields of application. Recent results have been especially impressive in image synthesis where learning the spatial appearance information is a key goal. This enables the generation of intermediate spatial data that corresponds to the original dataset. In the training stage, these models learn to decrease the distance of their output distribution to the actual data, and in the test phase, they map a latent space to the data space. Since these models have already learned their latent space mapping, one question is whether there is a function mapping the latent space to any aspect of the database for the given generator. In this work, it has been shown that that this mapping is relatively straightforward using small neural network models and by minimizing the mean square error. As a demonstration of this technique, two example use cases have been implemented: firstly, the idea to generate facial images with corresponding landmark data, and secondly generation of low-quality iris images (as would be captured with a smartphone user-facing camera) with a corresponding ground-truth segmentation contour.

Keywords: Generative models; Latent space mapping; Deep neural networks

2012 Elsevier Ltd. All rights reserved.

1. Introduction

Deep neural networks are a key driver of contemporary machine learning and artificial intelligence research and have begun to infiltrate the consumer world [1]. Advanced deep learning techniques are used to solve a wide range of long-standing problems in pattern recognition science. These approaches are famous for their power in designing and implementing regression and classification models.

Deep neural networks have shown great success when used as generative models. These models learn the distribution of a specific dataset and can generate new samples from the learned Probability Distribution Function (PDF). Classical methods include Variational Bayesian models, and Markov Chain Monte Carlo, which has been used to model the data distribution. Taking advantage of the neural networks non-linearity in defining the generative model goes back to early 2000 when [2] used *tanh*, the nonlinearity of a small neural network, in modelling the data distribution. These models are usually limited to small-sized problems due to the complexity, and they are also computationally prohibitive.

1.1. Deep Neural Networks as Generator

With the emergence of the low-cost, high-performance hardware for training deep neural networks, it became feasible to train and test big networks, and recently the generative models have also been taking advantage of deep neural networks in learning large size problems including image and sound generation.

In [3], the authors introduced two models to learn the data distribution. 1. PixelRNN which is composed of 12 2D Long Short Term Memory (LSTM) layers in which the LSTM units are applied in row and diagonal directions, namely RowLSTM and Diagonal BiLSTM respectively. 2. PixelCNN which is a fully convolutional deep neural network used to predict the conditional distribution at each pixel location. Since these models do not make use of latent space mapping, the framework proposed in this paper does not apply to them.

Variational Auto-Encoders (VAE) [4] are another approach for constructing a generative model. In this idea, the bottleneck of the auto-encoder network forms the latent space for the generative model. The encoder maps the input image to the latent space, and the decoder is a generator that maps the latent space into the sample space. The main difference between VAE and ordinary auto-encoders is the constraint on the latent space distribution. In VAE, the latent space is forced to obtain the Gaussian distribution by reducing the Kullback-Leibler (KL) divergence between the latent space distribution and the Gaussian. In practice, this is done by adding a term to the loss function to optimize the KL divergence in addition to the mean square error of the auto-encoder. The downside of this approach is that the network generates blurred images due to the mean square error loss [5].

Generative adversarial networks (GAN) presented by [6] are another type of generative model wherein two deep neural networks -a generator and a discriminator- are engaged in a min-max game. The generator accepts samples from the latent space with a uniform distribution. A deep neural network (the generator) converts this latent sample into a signal in the shape of the original dataset. The discriminator sees the signal from the generator and the original samples from the database and performs a binary classification task of whether it is drawn from the database or not. Authors in [6] show that the min-max loss function of GAN decreases the Jensen-Shanon Divergence (JSD) between the generator output and the data distribution.

1.2. Proposed Problem

A latent variable is a variable that is not observed but is used to describe the model or the observed data. Latent variables are said to exist in a ‘‘Latent space.’’ Generative models like VAE and GAN consider a latent variable for each data point and map the latent space to the data space by reducing the divergence between their output and data distribution. In this work a new problem is defined: by knowing the generator for a database, is there a function mapping the latent space onto any aspect of the database? (For the definition of ‘aspect’, see Table I) and if so, how can this mapping be defined? In this work, it has been shown that a deep neural network is able to accomplish this mapping, and the loss function can be as simple as mean square error, i.e., there is no need to enter the divergence into the objective of the mapping anymore. Solving such a problem has several applications, including consumer electronic design and data augmentation for regression problems. To the best of our knowledge, this is the first time such a problem has been considered in the literature.

The most relevant research work is that of the Gender Preserving Generative Adversarial Network (GP-GAN) [7] where adversarial networks are exploited to synthesize faces from the landmarks. The generator sub-network in GP-GAN is based on UNet [8] and DenseNet [9] architectures while the discriminator sub-network is based on [10]. Note that the network is using a new gender-preserving loss in parallel with the perceptual loss.

Another relevant study, Age-cGAN [11], employs a GAN to generate random faces and corresponding facial metadata. The focus of Age-cGAN is identity-preserving face aging where the person’s facial attributes are altered to age his/her face while the identity is preserved. The generator and discriminator sub-networks of Age-cGAN have the same architecture as [12]. This network can be used to synthesize augmented facial datasets incorporating aging of subjects.

The rest of the paper is organized as follows. Sec 2 presents the proposed method in detail. Two individual observations are illustrated in Sec 3 to generate random samples and their corresponding aspects. Finally, the conclusion and future works are presented in Sec 4.

2. Proposed Method

2.1. Problem Definition

The definitions of the phrases used in this work are given in Table I.

Table I. Phrases used throughout the paper, their definitions and symbols

Phrase	Definition	Symbol
Latent variable	A latent variable is a variable that is not observed but is used to describe the model or the observed data.	z
Latent space	Latent space is the space of the latent variable	Z -space
Aspect of dataset	Is the output of any local operation on the samples of the data	-
Perfect generator	A generator that gives the one-to-one correspondence between samples in the Z -space and every image in the database	$G()$
Perfect inverse of generator	A function that for each image in the database, gives the corresponding Z -space sample	$G^{-1}()$
Perfect aspect generator	Accepts the image and generates the perfect output for the specific aspect of the image	T
Aspect generator	Generates the data aspect directly from the latent space	$G_a()$
Deep neural sample generator	The generator from VAE, GAN, etc. Learns the data distribution and generates new samples. It is considered an approximation of the perfect generator.	$G^*()$

Problem definition: For a given database, if there is a perfect generator mapping a latent space into the data space -since the

generator is a local operation on the latent variable- by considering the data processing inequality, one can argue that the latent space includes all the information of the database. This indicates that any aspect of the dataset is extractable from the latent space alone. The problem considered in this work is to find a local operation that maps the latent space to the aspect space for a given generator and aspect. To the best of our knowledge, this is the first time such a problem has been defined. Solving this problem can facilitate high-speed implementation of regression solutions (in the presence of a real-time inverse of the generator,) which is a valuable solution in consumer electronic design. The other application will be data augmentation for regression problems. More discussion on applications of the proposed idea is given in section 4.

Naïve solutions: Suppose that there is a deep neural generator G^* mapping a uniformly distributed latent variable z_r^* to the data point x_r^* . This mapping is shown by $x_r^* = G^*(z_r^*)$. A new generator could be trained just for the specific data aspect, but training a new generator specifically for a feature set of the database would map an entirely different latent space (let's call it Z_l -space) to the feature space. It is not trivial to find the correspondence between the Z_r^* -space and Z_l -space. The other naïve solution would be to train a single model to learn the data and the feature distribution at the same time. This solution comes with complexities in implementation procedures. Since the feature and the data space might not be from the same class, and since the dimension and the objective function for each output could be totally different, the single model solution would not converge to a reasonable output.

Proposed solution: Inverse transform sampling theorem declares that by knowing the probability distribution of a random variable x_r , there is a transformation, mapping a uniformly distributed random variable z_r into the space of x_r . Since the generator network accepts a uniform random variable and transforms it to the image space, this network is learning an approximation of the data distribution. This provides justification for the ability of a neural network to learn the distribution of any aspect of the data for a given generator network. Our results firstly demonstrate that it is possible to train a network that learns the distribution of landmark annotations for a face generator, and secondly a network was trained that learns the segmentation for an iris generator.

Suppose that there is a perfect generator, mapping Z_r -space to the data space, $x_r = G(z_r)$. And the perfect inverse of the generator is shown by G^{-1} where for each image x_r in the database, the inverse of the generator gives the corresponding Z_r -space sample $z_{data} = G^{-1}(x_r)$.

The inverse of the generator is applied to all images in the database. The output space of the inverse of the generator is called Z_{data} -space which is a subspace of Z_r -space, but its distribution can be non-uniform. This Z_{data} -space and the perfect generator G contain all the information about the database, i.e., by knowing the Z_{data} -space and the perfect generator G , one can recreate the database. Since all the aspects of the database can be extracted using local operations on the data itself, considering the signal processing inequality, no further information is needed to produce any aspect of the database if Z_{data} -space and G are known. In fact, since G is a local operation, itself, any aspect of the database is extractable solely from knowing Z_{data} -space.

2.2. Mapping to the Aspect-Space

In this work, the aspect of a database is defined as the output of any local operation on the data samples. For example, the facial landmark detector gives an aspect (the landmark positions) of its input image, or an iris segmentor returns an aspect (the binary segmentation map) of its input iris image.

Suppose that for a given database there is a perfect aspect generator, T , which accepts the image x_{data} and generates the perfect output for the specific aspect of the image $T(x_{data})$. This generator can be a human making the facial landmark, or a super computer extracting features from an image, but if the aspect is simple enough to be estimated by a non-linear network G_a , the mean square error loss function between G_a and the perfect aspect generator T will be:

$$loss = \mathbb{E}_{z \sim p_{Z_{data}}(z)} \left\{ \left(G_a(z) - T(G(z)) \right)^2 \right\} \quad (1)$$

where $p_{Z_{data}}(z)$ is the probability distribution of Z_{data} . This loss function could be re-written as:

$$loss = \int_{z \sim p_{Z_{data}}(z)} p_{Z_{data}}(z) \left(G_a(z) - T(G(z)) \right)^2 dz \quad (2)$$

Since $p_{Z_{data}}(z)$ is positive for every z in Z_{data} -space, the minimum of this integral is zero and is achieved when:

$$G_a(z) = T(G(z)) \quad \forall z \text{ in } Z_{data}\text{-space} \quad (3)$$

The interesting part of the proposed method is that while the perfect generators on the right side of equation 3 are potentially very complicated, the aspect generator $G_a(z)$ can be small and simple. This is understandable by considering that the perfect aspect generator, T , needs to compress the information of the $G(z)$ and rule out the unnecessary information generated by the perfect generator, G , in order to produce the desired aspect. But on the left side of the equation the aspect generator, G_a , is bypassing all the complexity of T and G , and mapping the latent space Z_{data} -space to the aspect space. Adding any terms to reduce the divergence between the distribution of the aspect generator G_a , and perfect aspect generator T , is unnecessary since the latent space Z_{data} -space where the samples are drawn from is already learnt by the inverse of the generator.

In the following section, the results of the proposed method are presented for a face generator when the aspect is the facial landmarks and another observation is done for low-quality iris generation while the aspect is considered to be the binary map of the iris.

3. Results and Simulations

In this section, two different sets of data and various aspects of these datasets are presented. First, a generator is trained on CelebA [13] database, and the aspect of this database is a 49 point facial landmark [14] generated for each image. The second simulation is done on an augmented version of the Bath800 [15] and CASIA1000 [16] iris database, while the aspect is considered to be the iris segmentation map. These observations are described in detail in the following sections.

The Boundary Equilibrium Generative Adversarial Network (BEGAN) [17] scheme has been used to train the deep neural generator. In the BEGAN framework, the generator is similar to the generator in the original GAN method, but the discriminator is a deep auto-encoder, and the loss function tries to reduce the Wasserstein distance between the error of the auto-encoder for generated and original data. The reasons for selecting BEGAN are the simplicity of implementation, and the high-quality results of the generator. The BEGAN implementation is described in detail in Appendix A.

Since the generator learned the distribution of the database, it can map an N_z dimension random vector onto an interpolation point of the database distribution where N_z is the dimensionality of the latent space. The reverse applies as well. Having an image from the dataset, one can estimate the sample in the Z_r -space, which, when fed to the generator, will generate the image. The method used in this work is similar to one presented in [17] wherein the sample from the Z_r -space is approximated by optimizing the error function:

$$err = |x_r - G^*(z_r)| \quad (4)$$

where \mathbf{x}_r is the sample image and \mathbf{G}^* is the generator function. In all examples, the ADAM optimizer is used to solve the problem, with learning rate, β_1 , and β_2 equal to 0.1, 0.9, and 0.999, respectively. So the inverse of the generator accepts an image and produces the latent value for the given sample. If one applies the inverse of the generator to all the samples in the database, the output is a space of latent variables, which is a subset of \mathcal{Z}_r -space called \mathcal{Z}_{data} -space. As described before, this new space does not need to be uniformly distributed. One can also call this space the learnt latent space since it is derived from the inverse of the generator. In our experiments, this learned latent space is used to produce the aspect of the database.

3.1. Experiment 1: Face + landmarks

3.1.1. Database

The CelebA dataset [13] consisting of 202,599 original images with 40 unique attributes is used for training the GAN framework. The OpenCV frontal face cascade classifier [18] is used to detect facial regions, which are cropped and resized to 128×128 pixels. Initial landmark detection is performed using the method presented in [14] due to its ability to be effective on unconstrained faces. Authors in [14] have augmented the original cascade regression framework of [19] by proposing an incremental algorithm for cascade regression learning. This method personalizes the Supervised Descent Method (SDM) [20] for facial point localization, initializing the SDM offline on a large database of faces, and using newly tracked faces to update it incrementally. The detector in [14] uses a discriminative 3D facial deformable model fitted to the 2D image. The detector was trained on the 300W dataset [21]. It estimates a set of 49 landmarks defined by the contours of eyebrows, eyes, mouth and the nose as shown in Fig. 1.

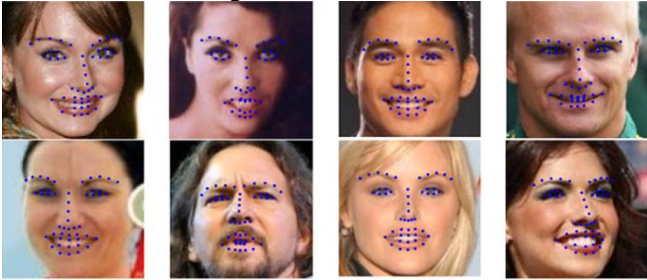


Fig. 1. Facial landmark detection from the discriminative deformable model [14]

3.1.2. Generating data, inverse of the generator and aspect mapping

Using the BEGAN framework, a generator has been trained on the CelebA database. The latent space is considered to be 64 dimensions. The ADAM optimizer was used with learning rates β_1 , and β_2 equal to 0.0001, 0.5, and 0.999, respectively. BEGAN was trained with the Lasagne library and Theano library in Python. A Geforce 1080ti desktop GPU was used to train the generator. Some randomly generated samples are shown in Fig 2.

The inverse of the generator (Equation 4) is then applied to all images in the dataset. All the outputs of the inverse of the generator make the \mathcal{Z}_{data} -space. Figure 3 shows how well the inverse of the generator works. In this figure, the estimated latent sample is fed to the generator and a very similar image is generated at the output.

Knowing all the samples in the \mathcal{Z}_{data} -space and also landmarks for each image (the output of perfect aspect generator T described in section 2.1,) an aspect generator is trained, mapping the \mathcal{Z}_{data} -space to the landmark space. See Fig 4.

The architecture of the Landmark Generator network trained to approximate the landmarks is shown in Table II. This network accepts 64-dimensional samples from the \mathcal{Z}_r -space and the

output is a set of 98 dimensions corresponding to 49 2D landmark points.



Fig. 2. Generating a random set of images using BEGAN framework on CelebA dataset.

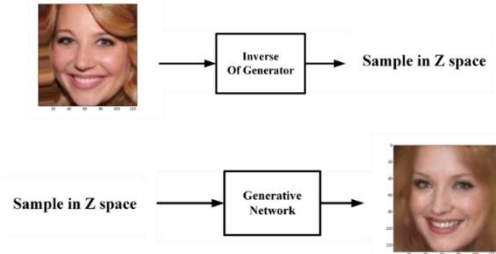


Fig. 3. The inverse of the generator can produce an accurate estimate of the latent value corresponding to each image in the database.

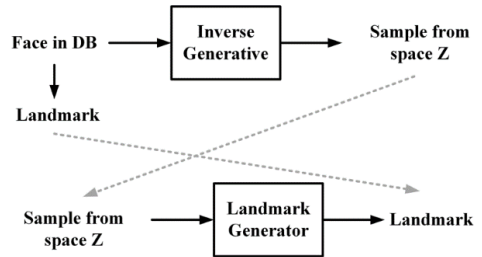


Fig. 4. Proposed method; the landmark generator maps \mathcal{Z} -space onto a “learned” landmark space

The loss function for this network is the Mean Square Error given by:

$$loss = \frac{1}{B_N \times 98} \sum_{k=1}^{B_N} \sum_{i=1}^{98} (o_i - t_i)^2 \quad (5)$$

wherein o_i is the i 'th output of the output layer, t_i is the i 'th target value, and B_N is the batch size which is set to 16. The ADAM optimizer is used to train the network with learning rates β_1 , and β_2 equal to 0.0003, 0.9, and 0.999 respectively. The training was done for 1000 epochs using all data. No validation and test set were used in the method.

Adding validation and test sets made the results less accurate, since after reducing the number of samples in the training set, the network was blind to some examples and could not reconstruct the landmark distribution accurately. To the best of our knowledge, this is the first attempt to generate samples and their corresponding landmarks at the same time.

In the feedforward step shown in Fig 5, a uniformly distributed random vector is fed concurrently into the Generator (from BEGAN) and the Landmark Generator. The first generates a random interpolated face, and the Landmark Generator

provides a 2D set of landmarks corresponding to the generated face.

Table II. The architecture of the landmark generator. It is a small, fully connected deep neural network

Layer name	Layer kind	Number of nodes	Activation
Input Layer	Input	64	---
First Hidden	Fully connected	128	RELU
Second Hidden	Fully connected	128	RELU
Third Hidden	Fully connected	128	RELU
Output Layer	Fully connected	98	Sigmoid

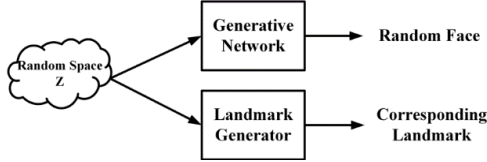


Fig. 5. Feedforward of the proposed method. The uniformly distributed random vector is fed to the Generative network given by BEGAN, and the Landmark Generator from the proposed method produces the landmark positions for the generated face image

Fig. 6 shows example results. Initial results show the Landmark Generator has learned to map a set of landmark points from the same Z_r -space as the facial generator, with good generalization across varying pose & illumination conditions.

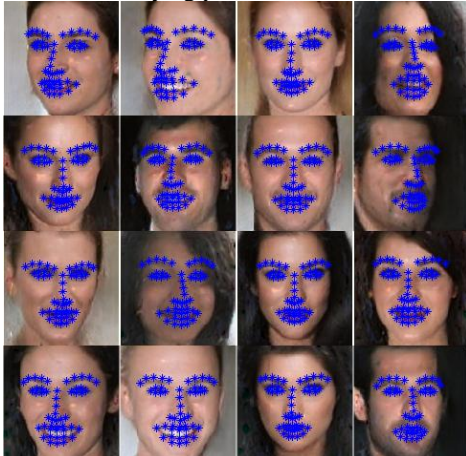


Fig. 6. Random generated faces and their corresponding landmarks.

In [17] the authors investigate the continuity of the face distribution given by the BEGAN face generator by feeding it a random set of numbers, interpolating two samples in Z_r -space, and observing the gradual changes from one face to another. To investigate the continuity of the landmark estimator distribution, the same approach is used. The z_r vectors for a given image and its mirror image are estimated using the inverse generator method, and 14 interpolation points between the two Z_r -space samples are fed into the face generator and Landmark Generator networks. One example result from this experiment is shown in Fig. 7. This figure shows that the landmark distribution estimated by the Landmark Generator network is smooth in the Z_r -space. More results, including illumination variation and pose variation are presented in a video¹ that is generated by smoothly moving the latent variable to make the face and landmarks.

3.2. Experiment 2: Iris+segmentation

3.2.1. Dataset

In this experiment, two iris databases, Bath800 and CASIA1000, have been used to learn the generator. Bath800 is made of 31,997 iris images with a resolution of [1280×960] taken from 800 individuals, and CASIA1000 has 20,000 Near Infrared images with a resolution of [640×480]. All these images are resized to [128×96] in this experiment. None of these

databases are provided with ground truth, but since they are high-quality datasets taken in highly constrained conditions, any industry standard segmentation tool will give a reasonably high-quality segmentation result. In this work, the segmentation provided by a commercial iris segmentation tool (MIRLIN [22]) is treated as the ground truth for the segmentation task. To increase the number of samples in the database, several augmentation techniques have been applied to the original dataset, including contrast reduction inside and outside of the iris region, and adding shadow and motion blur. For a detailed description of the augmentation process, see [23]. Some samples of the database after applying the augmentation, and their corresponding ground truth maps are shown in Fig 8.



Fig. 7. Interpolating in the Z_r -space between a face and its mirror image indicates strong continuity for both BEGAN and the landmark generator.

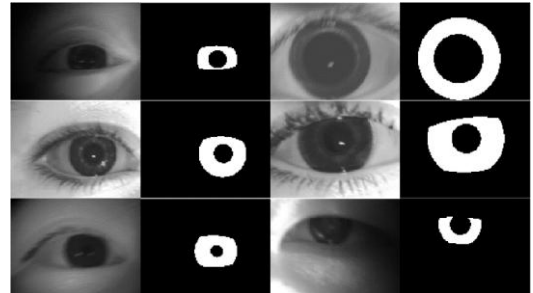


Fig. 8. Iris samples after applying augmentation and their corresponding segmentation map.

After the augmentation process, there are 262K samples in our training set. This database is designed initially to train a deep neural network, segmenting low-quality iris images [23]. In this section, it is used to show that the aspect of the dataset could be something more than points or features of the image. In fact, it can be the same size as the image like a binary map.

3.2.2. Generating data, inverse of the generator and aspect mapping

The method to train the deep neural generator is exactly similar to the previous experiment. The BEGAN scheme has been used to learn the data distribution for the low-quality iris database. The only difference is the size of the images where, in the previous experiment, the face images were 128×128, but in this experiment, the images are 128×96. The latent space is 64 dimensions. The ADAM optimizer was used with learning rates β_1 , and β_2 , equal to 0.0001, 0.5, and 0.999, respectively. The BEGAN model was trained with the Lasagne library and Theano library in Python. A Geforce 1080ti desktop GPU was used to train the generator. Some randomly generated samples are shown in Fig 9. The next step is to apply the inverse of the generator to all samples in the dataset. The inverse of the generator accepts an

¹ <https://youtu.be/PWdT3Q5T5U8>

image and estimates its corresponding latent space sample. After applying the inverse of the generator to all samples in the dataset, all samples will be in \mathcal{Z}_{data} -space.

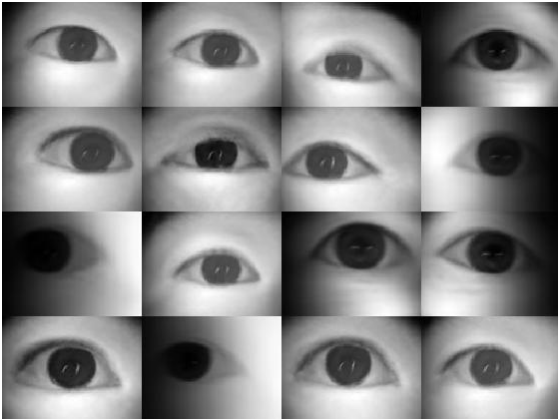


Fig. 9. The random samples drawn from the generator trained using the BEGAN method on the low-quality iris database

In this experiment, the aspect of the dataset is the binary segmentation map of the iris image. The perfect aspect generator T is the iris segmentation tool (MIRLIN [22]), described in the previous section. The architecture of the aspect generator network, (mapping the latent space to the aspect space) is exactly similar to the generator in BEGAN, as described in Appendix A.

The loss function for the aspect generator is the mean square error given by:

$$loss = \frac{1}{B_N \times 96 \times 128} \sum_{k=1}^{B_N} \sum_{i=1}^{96} \sum_{j=1}^{128} (o_{ij} - t_{ij})^2 \quad (6)$$

where o_{ij} is the (i, j) 'th pixel of the output layer, t_{ij} is the (i, j) 'th pixel of the target (iris segmentation map), and B_N is the batch size, which is set to 16. The ADAM optimizer is used to train the network with the learning rates β_1 , and β_2 , equal to 0.0003, 0.9, and 0.999 respectively. The training was done for 1000 epochs using all data. No validation and test set were used in the method.

The feedforward model is shown in Fig 10.

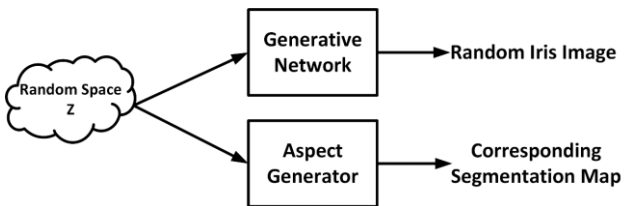


Fig. 10. The feedforward model for generating a random low-quality iris image and its corresponding segmentation map.

The results of this experiment are shown in Fig 11. In this figure, the iris image is generated by the generator trained in the BEGAN method, and the segmentation maps are generated using the proposed aspect generation method.

In order to investigate the continuity of the mapping for both the iris and the segmentation, a video² illustrates the smooth changes in both the iris samples and the segmentation. The sequences are created by smoothly moving the latent sample in the \mathcal{Z}_r -space.

4. Conclusion and Future Works

One of the most amazing applications of the deep neural networks is to learn the data distribution and draw new samples from the learned distribution. VAE and GAN are two successful

implementations of such an application. In each of these methods, the objective is to reduce the divergence of the generator output and the original data. These methods also take advantage of using a latent space that gives an opportunity to manipulate data and also learn any aspect of the database straight from the latent space.

After training the generator, it can be considered a deterministic local nonlinear operation that maps the latent space onto the data space. Considering the data processing inequality, any local operation on the data set cannot inject extra information to the data. This means that all information of the individual samples is already in the latent space. This explains why any aspect of the database can be extracted straight from the latent space.

In this work, it has been shown that the previous statement is true, and different aspects of the database, landmarks for the face and segmentation map of an iris image can be mapped from the latent space.

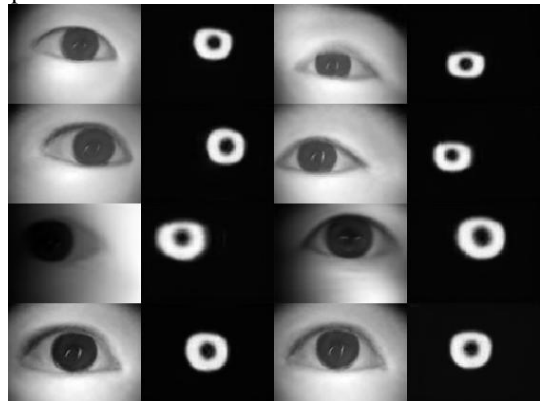


Fig. 11. Randomly generated low-quality iris images and their corresponding segmentation maps. First and third columns are randomly generated iris images; second and fourth columns are their corresponding segmentation maps created by the proposed framework.

There are several applications for the presented framework. Since the latent space is smaller than the actual data space, this method can be used as a compression method. The feature extraction from latent space is fast, which is useful in designing fast consumer electronic devices. The other application is using the generative models as an augmentation technique. Data augmentation is a crucial step for modern machine learning frameworks, including deep learning approaches. New deep neural networks need a large number of samples to be trained in order to avoid overfitting. The augmentation process introduces a certain amount of uncertainty into the database, which helps the network prevent overfitting and generalizes the results. To the best of our knowledge, augmentations presented for regression problems are all applied in the image space. These operations include flipping, rotating, manipulating the contrast and illumination of the image, and applying distortions to the image. The framework presented in this work can utilize the data and ground truth generation in latent space. Our observations show that the mapping for both the data generator and aspect generator is continuous and smooth in the latent space. This gives the opportunity to generate a large number of samples and their corresponding ground truth, thus expanding the database by introducing more variations through the generation of multiple intermediate samples. Future work will include the investigation of the influence of the generative augmentation technique in training regression networks.

Acknowledgments

The authors would like to thank Joseph Lemley and Kimberly Sowell for their helpful comments.

² <https://youtu.be/BY9cVZPmgRU>

The research work presented here was funded under the Strategic Partnership Program of Science Foundation Ireland (SFI) and co-funded by SFI and FotoNation Ltd. Project ID: 13/SPP/I2868 on “Next Generation Imaging for Smartphone and Embedded Platforms”.

Portions of the research in this paper use the CASIA-IrisV4 collected by the Chinese Academy of Sciences' Institute of Automation (CASIA).

References

- [1] J. Lemley, S. Bazrafkan, P. Corcoran, Deep Learning for Consumer Devices and Services: Pushing the limits for machine learning, artificial intelligence, and computer vision., IEEE Consum. Electron. Mag. 6 (2017) 48–56. doi:10.1109/MCE.2016.2640698.
- [2] H. Valpola, J. Karhunen, An unsupervised ensemble learning method for nonlinear dynamic state-space models, Neural Comput. 14 (2002) 2647–2692.
- [3] A. van den Oord, N. Kalchbrenner, K. Kavukcuoglu, Pixel recurrent neural networks, arXiv Prepr. arXiv1601.06759. (2016).
- [4] D.P. Kingma, M. Welling, Auto-encoding variational bayes, arXiv Prepr. arXiv1312.6114. (2013).
- [5] K. Frans, Variational Autoencoders Explained, (2016).
- [6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Adv. Neural Inf. Process. Syst., 2014: pp. 2672–2680.
- [7] X. Di, V.A. Sindagi, V.M. Patel, GP-GAN: Gender Preserving GAN for Synthesizing Faces from Landmarks, arXiv Prepr. arXiv1710.00962. (2017).
- [8] O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, in: N. Navab, J. Hornegger, W.M. Wells, A.F. Frangi (Eds.), Med. Image Comput. Comput. Interv. -- MICCAI 2015 18th Int. Conf. Munich, Ger. Oct. 5-9, 2015, Proceedings, Part III, Springer International Publishing, Cham, 2015: pp. 234–241. doi:10.1007/978-3-319-24574-4_28.
- [9] G. Huang, Z. Liu, L. v. d. Maaten, K.Q. Weinberger, Densely Connected Convolutional Networks, in: 2017 IEEE Conf. Comput. Vis. Pattern Recognit., 2017: pp. 2261–2269. doi:10.1109/CVPR.2017.243.
- [10] P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, arXiv Prepr. arXiv1611.07004. (2016).
- [11] G. Antipov, M. Baccouche, J.-L. Dugelay, Face Aging With Conditional Generative Adversarial Networks, arXiv Prepr. arXiv1702.01983. (2017).
- [12] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv Prepr. arXiv1511.06434. (2015).
- [13] Z. Liu, P. Luo, X. Wang, X. Tang, Deep Learning Face Attributes in the Wild, in: 2015 IEEE Int. Conf. Comput. Vis., 2015: pp. 3730–3738. doi:10.1109/ICCV.2015.425.
- [14] A. Asthana, S. Zafeiriou, S. Cheng, M. Pantic, Incremental Face Alignment in the Wild, in: 2014 IEEE Conf. Comput. Vis. Pattern Recognit., 2014: pp. 1859–1866. doi:10.1109/CVPR.2014.240.
- [15] S. Rakshit, Novel methods for accurate human iris recognition, University of Bath, 2007.
- [16] CASIA Iris Image Database, (2010).
- [17] D. Berthelot, T. Schumm, L. Metz, Began: Boundary equilibrium generative adversarial networks, arXiv Prepr. arXiv1703.10717. (2017).
- [18] OpenCV, Face Detection using Haar Cascades, (n.d.).
- [19] X. Cao, Y. Wei, F. Wen, J. Sun, Face alignment by Explicit Shape Regression, in: 2012 IEEE Conf. Comput. Vis. Pattern Recognit., 2012: pp. 2887–2894. doi:10.1109/CVPR.2012.6248015.
- [20] X. Xiong, F.D. la Torre, Supervised Descent Method and Its Applications to Face Alignment, in: 2013 IEEE Conf. Comput. Vis. Pattern Recognit., 2013: pp. 532–539. doi:10.1109/CVPR.2013.75.
- [21] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, M. Pantic, 300 Faces in-the-Wild Challenge: The First Facial Landmark Localization Challenge, in: 2013 IEEE Int. Conf. Comput. Vis. Work., 2013: pp. 397–403. doi:10.1109/ICCVW.2013.59.
- [22] F. Inc., MIRLIN, (2015).
- [23] S. Bazrafkan, S. Thavalengal, P. Corcoran, An End to End Deep Neural Network for Iris Segmentation in Unconstraint Scenarios, arXiv Prepr. arXiv1712.02877. (2017).
- [24] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), CoRR. abs/1511.0 (2015).

Appendices

A. Boundary Equilibrium Generative Adversarial Networks

In this work, the Boundary Equilibrium Generative Adversarial Network (BEGAN), presented in [17] is implemented to train a generator. In this approach, the discriminator network is an auto-encoder, and the generator architecture is the same as the decoder part of the discriminator. The encoder and decoder parts are shown in Fig A.1 and Fig A.2 respectively.

In the encoder, all kernels are 3×3 , and ELU [24] nonlinearity is used in all layers apart from the red layers where the kernel size is 1×1 , and no nonlinearities are employed. Also no non-linearity is applied to the fully connected layers. In the decoder network, all convolutional layers have 64 channels, while in the encoder, the number of the channels is gradually increased to 128, 192, and 256 after each pooling layer.

Suppose that x is the real data coming from the database, z is a sample from the uniformly distributed random space \mathcal{Z} , \mathcal{D} is the auto-encoder function with the loss defined by:

$$\mathcal{L}(v) = |v - \mathcal{D}(v)|^2 \quad (\text{A.1})$$

where v is the input to the auto-encoder. The objectives for BEGAN given by [17] are:

$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t \cdot \mathcal{L}(G(z_D)) & \text{for } \theta_D \\ \mathcal{L}_G = \mathcal{L}(G(z_G)) & \text{for } \theta_G \\ k_{t+1} = k_t + \lambda_k (\gamma \mathcal{L}(x) - \mathcal{L}(G(z_G))) & \text{for each training step } t \end{cases} \quad (\text{A.2})$$

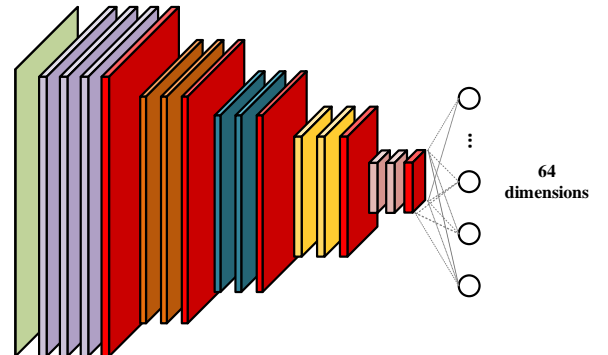


Fig A.1. Encoder architecture for BEGAN

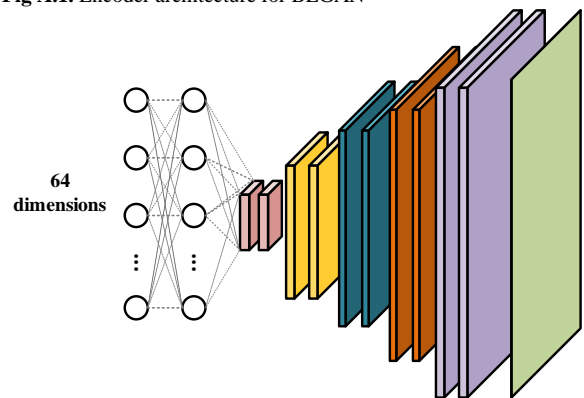


Fig A.2. Decoder architecture for BEGAN

where \mathcal{L}_D is the discriminator loss, \mathcal{L}_G is the generator loss, $G(v)$ is the output of the generator for input vector v , γ is the equilibrium hyper parameter set to 0.5 in this work, and λ_k is the learning rate for k . The ADAM optimizer is used with learning rates β_1 , and β_2 , equal to 0.0001, 0.5, and 0.999, respectively. BEGAN was trained with the Lasagne library on the top of Theano library in Python.

Appendix G

Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN)

Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN)

Shabab Bazrafkan

Dept. Electrical & Electronic Engineering, College of Engineering & Informatics
National University of Ireland Galway
s.bazrafkan1@nuigalway.ie

Hossein Javidnia

Dept. Electrical & Electronic Engineering, College of Engineering & Informatics
National University of Ireland Galway
h.javidnia1@nuigalway.ie

Peter Corcoran

Dept. Electrical & Electronic Engineering, College of Engineering & Informatics
National University of Ireland Galway
peter.corcoran@nuigalway.ie

Abstract

One of the most interesting challenges in Artificial Intelligence is to train conditional generators which are able to provide labeled adversarial samples drawn from a specific distribution. In this work, a new framework is presented to train a deep conditional generator by placing a classifier in parallel with the discriminator and back propagate the classification error through the generator network. The method is versatile and is applicable to any variations of Generative Adversarial Network (GAN) implementation, and also gives superior results compared to similar methods.

1 Introduction

Deep Learning influences almost every aspect of the machine learning and artificial intelligence. It gives superior results for classification, and regression problems compare to classical machine learning approaches [6]. The other impact of Deep Learning is on generative models [5]. In this work, the problem of conditional generators is considered, and a global solution is presented. The conditional generative models are models which can generate a class-specific sample given the right latent input. As one example, these generators can learn the data distribution for male/female faces and produce outputs that match a single (male/female) class. Several researchers have attempted to provide a solution to this problem [8, 9]. But none of them are able to propose a global solution. In [8] the authors introduce a variation of GAN known as conditional GAN, wherein the model is similar to the ordinary GAN, but the latent space is conditional with respect to the class label. This approach is versatile enough to be extended to other GAN variations, but there is no mathematical proof that the trained generator is able to provide distinct samples for different classes. In our experiments, applying this method to the BEGAN [2] scheme to generate male/female images did not generate gender-specific samples. This is explained in more detail in section 3. The most successful implementation of the class specified generative model is Auxiliary Classifier GANs (ACGAN) [9] wherein by adding a classification term to the generator and discriminator loss, the generator is forced to generate a specific class of data for a given input. see figure 1a.

Preprint. Work in progress.

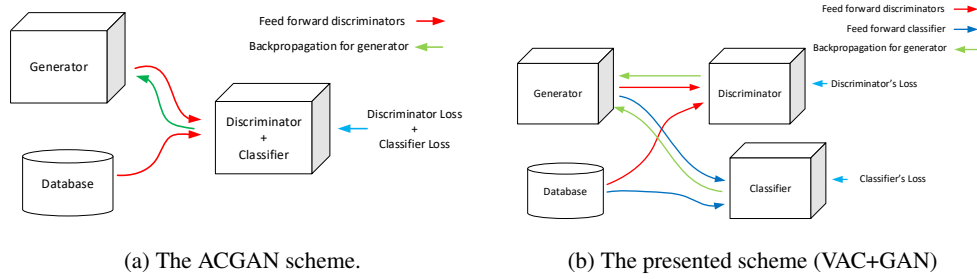


Figure 1: ACGAN vs presented model.

The main problem with ACGAN is that it is not versatile enough to be applied to other GAN variations. Mixing the loss of discriminator and the classifier will alter the training convergence specially if the output of the discriminator is from a different type compare to the classifier's output. For example in the BEGAN implementation, the output of the discriminator is an image (2D matrix) compare to the output of the classifier which is a (1D) vector. Merging the loss for two different output types into a single loss alters the convergence of the network.

In this work, a new approach for training a class specified generator model is presented which is independent of the generator and discriminator structure. i.e., the presented method can be applied to any model that is already converging. The method is called Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN). The mathematical proof for the effectiveness of the method is also presented. The idea is to get the classification term (in ACGAN) out of the discriminator's loss function by adding a classifier network that back-propagates through the generator.

In the next section, the proposed idea is presented alongside with the mathematical proof of the effectiveness of the method, the experimental results are given in section three and conclusions are presented in the last section.

2 Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN)

The concept proposed in this research is to place a classifier network in parallel with the Discriminator. The classifier accepts the samples from the generator, and the classification error is back-propagated through the classifier and the generator. The model structure is shown in figure 1b.

In this section, the proposed method is investigated for two class problems. In this case, the classifier is a binary classifier with binary cross-entropy loss function. The notations used in the mathematical proof are as follows:

1. $V(G, D)$: is the objective function for a general generative model, wherein G and D are Generator and Discriminator.
2. The latent space Z is partitioned into Z_1, Z_2 subsets. This means that Z_1 and Z_2 are disjoint and their union is equal to the Z -space.
3. C is the classifier function.
4. \mathcal{L}_{ce} is the binary cross-entropy loss function.

Proposition 1. For a fixed Generator and Discriminator, the optimal Classifier is

$$C_{G,D}^* = \frac{p_{X_1}(\mathbf{x})}{p_{X_1}(\mathbf{x}) + p_{X_2}(\mathbf{x})} \quad (1)$$

wherein $C_{G,D}^*$ is the optimal classifier, and $p_{X_1}(\mathbf{x})$, and $p_{X_2}(\mathbf{x})$ are the distributions of generated samples for the first and second class respectively.

Proof. The objective function for the model is given by:

$$O(G, D, C) = V(G, D) + \mathcal{L}_{ce}(C) \quad (2)$$

This can be rewritten as

$$O(G, D, C) = V(G, D) - \mathbb{E}_{\mathbf{z} \sim p_{Z_1}(\mathbf{z})} [\log(C(G(\mathbf{z})))] - \mathbb{E}_{\mathbf{z} \sim p_{Z_2}(\mathbf{z})} [\log(1 - C(G(\mathbf{z})))] \quad (3)$$

which is given by

$$O(G, D, C) = V(G, D) - \left\{ \int p_{Z_1}(\mathbf{z}) \log(C(G(\mathbf{z}))) + p_{Z_2}(\mathbf{z}) \log(1 - C(G(\mathbf{z}))) d\mathbf{z} \right\} \quad (4)$$

Considering $G(z_1) = x_1$ and $G(z_2) = x_2$ we get

$$O(G, D, C) = V(G, D) - \left\{ \int p_{X_1}(\mathbf{x}) \log(C(\mathbf{x})) + p_{X_2}(\mathbf{x}) \log(1 - C(\mathbf{x})) d\mathbf{x} \right\} \quad (5)$$

The function $f \rightarrow m \log(f) + n \log(1-f)$ reaches its maximum at $\frac{m}{m+n}$ for any $(m, n) \in \mathbb{R}^2 \setminus \{0, 0\}$, concluding the proof. \square

Theorem 1. *The maximum value for $\mathcal{L}_{ce}(C)$ is $\log(4)$ and is achieved if and only if $p_{X_1} = p_{X_2}$.*

Proof. For $p_{X_1} = p_{X_2} \implies C_{G,D}^* = \frac{1}{2}$ and by observing that

$$-\mathcal{L}_{ce}(C) = \mathbb{E}_{\mathbf{x} \sim p_{X_1}(\mathbf{x})} (\log(C(\mathbf{x}))) + \mathbb{E}_{\mathbf{x} \sim p_{X_2}(\mathbf{x})} (\log(1 - C(\mathbf{x}))) \quad (6)$$

results in

$$\mathcal{L}_{ce}(C_{G,D}^*) = -\log\left(\frac{1}{2}\right) - \log\left(\frac{1}{2}\right) = \log(4) \quad (7)$$

To show that this is the maximum value, from equation 5 we have

$$\mathcal{L}_{ce}(C_{G,D}^*) = - \int p_{X_1}(\mathbf{x}) \log\left(\frac{p_{X_1}(\mathbf{x})}{p_{X_1}(\mathbf{x}) + p_{X_2}(\mathbf{x})}\right) d\mathbf{x} - \int p_{X_2}(\mathbf{x}) \log\left(\frac{p_{X_2}(\mathbf{x})}{p_{X_1}(\mathbf{x}) + p_{X_2}(\mathbf{x})}\right) d\mathbf{x} \quad (8)$$

which is equal to

$$\mathcal{L}_{ce}(C_{G,D}^*) = \log(4) - \int p_{X_1}(\mathbf{x}) \log\left(\frac{p_{X_1}(\mathbf{x})}{\frac{p_{X_1}(\mathbf{x}) + p_{X_2}(\mathbf{x})}{2}}\right) d\mathbf{x} - \int p_{X_2}(\mathbf{x}) \log\left(\frac{p_{X_2}(\mathbf{x})}{\frac{p_{X_1}(\mathbf{x}) + p_{X_2}(\mathbf{x})}{2}}\right) d\mathbf{x} \quad (9)$$

results in

$$\begin{aligned} \mathcal{L}_{ce}(C_{G,D}^*) &= \log(4) - KL\left(p_{X_1}(\mathbf{x}) \left\| \frac{p_{X_1}(\mathbf{x}) + p_{X_2}(\mathbf{x})}{2} \right.\right) \\ &\quad - KL\left(p_{X_2}(\mathbf{x}) \left\| \frac{p_{X_1}(\mathbf{x}) + p_{X_2}(\mathbf{x})}{2} \right.\right) \end{aligned} \quad (10)$$

Where KL is the Kullback-Leibler divergence, which is always positive or equal to zero, concluding the proof. \square

Theorem 2. *Minimizing the binary cross-entropy loss function \mathcal{L}_{ce} for the classifier C is increasing the Jensen-Shannon divergence between p_{X_1} and p_{X_2} .*

Proof. the Jensen-Shannon divergence between p_1 and p_2 is given by

$$JSD(p_1 || p_2) = \frac{1}{2} KL\left(p_1 \left\| \frac{p_1 + p_2}{2} \right.\right) + \frac{1}{2} KL\left(p_2 \left\| \frac{p_1 + p_2}{2} \right.\right) \quad (11)$$

considering equation 10 and 11, it gives

$$\mathcal{L}_{ce}(C_{G,D}^*) = \log(4) - 2JSD(p_{X_1} || p_{X_2}) \quad (12)$$

minimizing \mathcal{L}_{ce} is equal to maximizing $JSD(p_{X_1} || p_{X_2})$, concluding the proof. \square

Here it has been shown that placing the classifier C and add its loss value the generative framework pushes the generator to increase the distance of samples that are drawn from a specific class with respect to the other class. For example, in the male/female face scenario, one can use a partition of Z space to generate male and another partition to generate female samples.

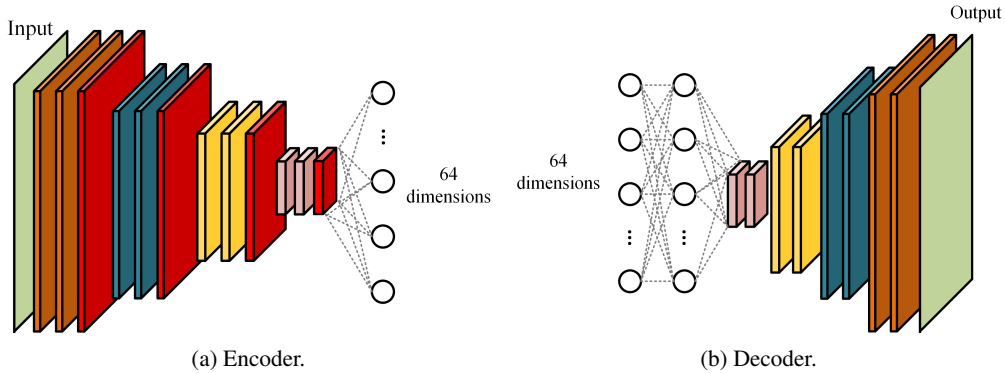


Figure 2: Encoder and Decoder architectures used in BEGAN approach.

3 Experimental Results

In this section, an experiment is conducted to show the effectiveness of the proposed scheme while different measures are used to show the diversity of the generated samples including Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Universal Quality Index (UQI), and Structural Similarity Index (SSIM). These measurements are explained in Appendix A. MSE, RMSE and MAE show the difference between two images. The higher values for these metrics correspond to higher variation of the generated images. UQI and SSIM measure the structural similarity between two samples. Lower value for these measurements correspond to less similarity. In evaluating generative models higher values for MSE, RMSE, and MAE and lower values for UQI and SSIM is desirable.

In this section, all the networks are trained in Lasagne [4] on top of Theano [1] library in Python.

The experiment is conducted by training a gender specified generator using the BEGAN [2] structure trained on CelebA database. The results of the proposed method are compared against the results of conditional GAN idea applied to the BEGAN framework. The comparisons with ACGAN method is not available since applying this method to BEGAN framework altered the convergence of the model and the generator constrained to a deterministic output even after the first epoch.

The CelebA dataset [7] consisting of 202,599 original images is used for training our GAN framework. The OpenCV frontal face cascade classifier [10] is used to detect facial regions which are cropped and resized to 48×48 pixels. In the BEGAN framework the generator network is a typical GAN generator which has the same architecture as the decoder part of an auto-encoder. The network used in our experiment contains one fully connected layer which maps the input to a 3D layer. Next layers are all convolutional layers followed by (2, 2) un-pooling layers for every second convolution. The exponential linear unit (ELU) [3] is used as activation function except in the last layer wherein no non-linearity has been applied. And the discriminator network is an auto-encoder. The input of the auto-encoder is the image (48×48). The encoder part of the network is made of convolutional layers with ELU activation function. The downscaling in these layers is obtained by using (2, 2) stride in every second convolutional layer. The architecture of decoder is the same as the generator network. And the bottleneck of the auto-encoder is a fully connected layer with no activation function. The encoder and decoder networks used for training the BEGAN are shown in figures 2a and 2b respectively. The layers shown in red apply no nonlinearity to the data.

The loss function for training the conditional BEGAN (CBEGAN) is given by:

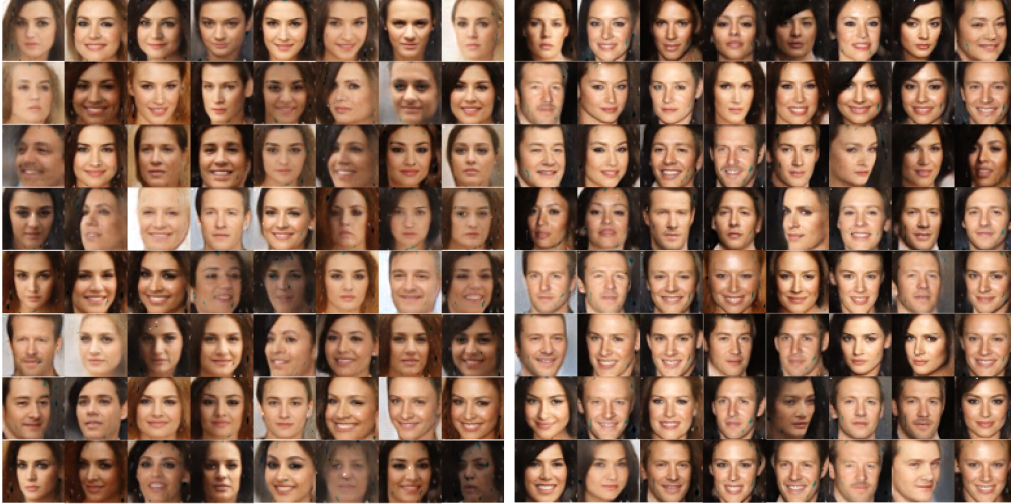
$$\begin{aligned}
 L_d &= L(x) - k_t \cdot L(G(z|c)) \\
 L_g &= L(G(z|c)) \\
 k_{t+1} &= k_t + \lambda_k (\gamma L(x) - L(G(z|c)))
 \end{aligned} \tag{13}$$

Where L_g and L_d are generators and discriminators losses respectively. G is the generator function, z is a sample from the latent space, c is the class label, x is the sample drawn from the database, λ_k is the learning rate for k , γ is the equilibrium hyper parameter set to 0.5 in this work, and L is the auto-encoders loss defined by

$$L(v) = |v - D(v)|^2 \tag{14}$$

Table 1: the classifier structure for the CelebA+BEGAN experiment.

Layer	Type	kernel	Activation
Input	Input(48 × 48)	–	–
Hidden 1	Conv	3 × 3(16 ch)	ReLU
Pool 1	Max pooling	2 × 2	–
Hidden 2	Conv	3 × 3(8 ch)	ReLU
Pool 2	Max pooling	2 × 2	–
Hidden 3	Dense	1024	ReLU
Output	Dense	1	Sigmoid



(a) Constrained to generate female samples.

(b) Constrained to generate male samples.

Figure 3: Generator trained using CBEGAN method.

The proposed method needs a classifier to back-propagate the classification error throughout the generator. The classifier used in this experiment is a simple deep classifier given in table 1. The loss functions used to train the VAC+GAN applied to BEGAN is given by

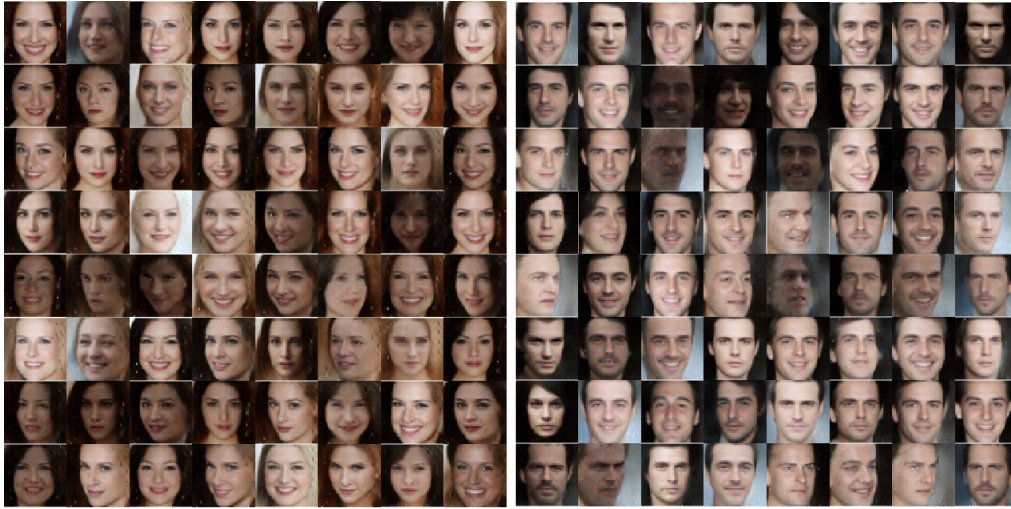
$$\begin{aligned}
 L_d &= L(x) - k_t \cdot L(G(z|c)) \\
 L_g &= \vartheta \cdot L(G(z|c)) + \zeta \cdot BCE \\
 k_{t+1} &= k_t + \lambda_k (\gamma L(x) - L(G(z|c)))
 \end{aligned} \tag{15}$$

where BCE is the binary cross-entropy loss of the classifier, and ϑ and ζ are set to 0.997 and 0.003 respectively. The optimizer used for training the generator and discriminator is ADAM with learning rate, β_1 and β_2 equal to 0.0001, 0.5 and 0.999 respectively. And the classifier is optimized using nestrov momentum gradient descent with learning rate and momentum equal to 0.01 and 0.9 respectively.

The latent space has 64 dimensions and the first dimension is used to partition the latent space in two subspaces corresponding to two classes. The results for the CBEGAN and proposed method are shown in figures 3 and 4.

As it is shown in these figures, the gender-specific generator fails to correctly generate samples for a specific class when the conditional GAN is applied. But the proposed method is able to correctly constrain the generator to make samples drawn from a specific class. In order to compare the models, 80 random male and 80 random female samples have been generated using the trained generators. Three observations have been conducted on these samples:

1. Each male sample has been compared to all the other male samples, and all the metrics have been calculated for these comparisons, and the average of these numbers has been obtained (blue bars).



(a) Constrained to generate female samples.

(b) Constrained to generate male samples.

Figure 4: Generator trained using the proposed method (VAC+GAN).

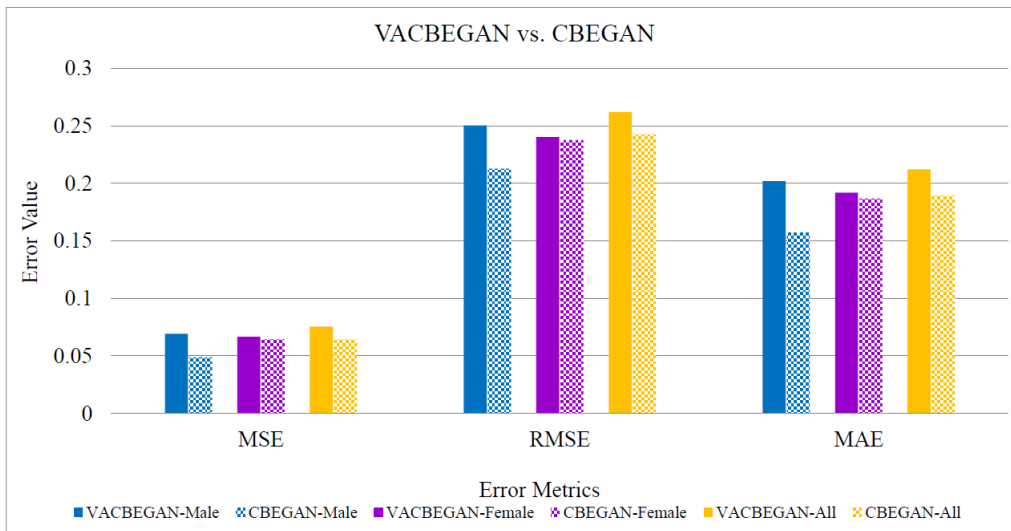


Figure 5: UQI and SSIM metrics for presented method vs. CBEGAN. Lower values show higher performance.

2. Each female sample has been compared to all the other female samples, and all the metrics have been calculated for these comparisons, and the average of these numbers has been obtained (purple bars).
3. Each male samples has been compared to all female samples, and all the metrics have been calculated for these comparisons, and the average of these numbers has been obtained (yellow bars).

The aforementioned measurements are illustrated in figures 5 and 6 for the CBEGAN and proposed method. The lower value of UQI and SSIM shows the less similarity between samples. In figure 5, from the first two observations (blue and purple bars) it is shown that the proposed method is able to generate samples in each class that are not similar. From the third observation (yellow bars) it is shown that the inter-class similarity in the proposed method is less than CBEGAN i.e., This shows that the generated samples from different classes are less similar to each other. The higher value

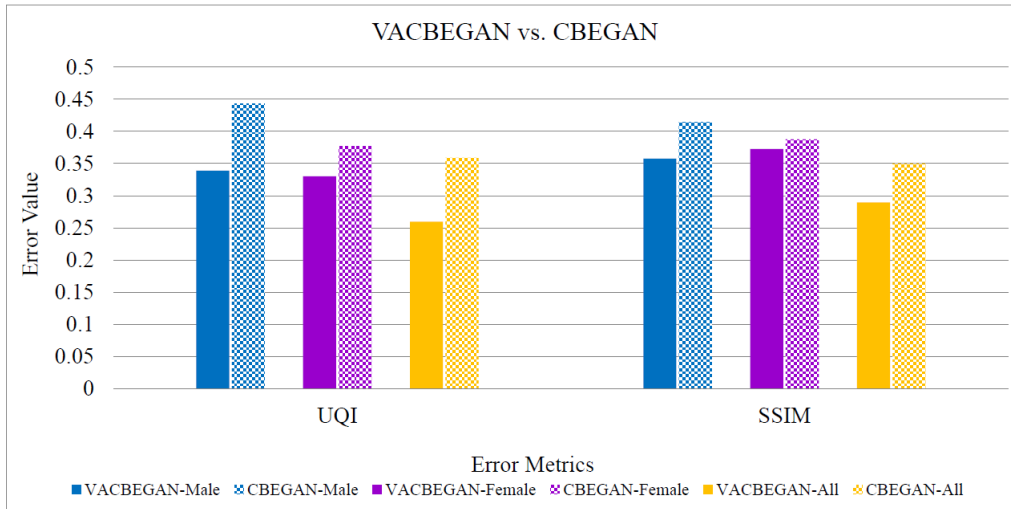


Figure 6: MSE, RMSE and MAE metrics for presented method vs. CBEGAN. higher values show better performance.

of MSE, RMSE, and MAE shows the higher variation of the generated images. As it is shown in figure 6 the proposed method is able to generate a higher variation of samples for each class and also between classes.

4 Discussion and Conclusion

In this work a new approach has been introduced to train conditional deep generators. In this work it has been proven that VAC+GAN is applicable to any GAN framework regardless of the model structure and/or loss function (see Section 2). The idea is to place a classifier in parallel to the discriminator network and back-propagate the loss of this classifier through the generator network in the training stage. It has also been shown that the presented framework increases the Jensen Shannon Divergence (JSD) between classes generated by the deep generator. i.e., the generator can produce samples drawn from a desired class. The results has been compared to another versatile method known as Conditional GAN (CGAN) for gender specified face generation.

The future work includes applying the method to datasets with large number of classes and also extend the implementation for bigger size images. Other idea is to extend the current approach to regression problems. This can help to generate samples with specific continuous aspect.

Acknowledgments

This research is funded under the SFI Strategic Partnership Program by Science Foundation Ireland (SFI) and FotoNation Ltd. Project ID: 13/SPP/I2868 on Next Generation Imaging for Smartphone and Embedded Platforms.

References

- [1] James Bergstra, Frédéric Bastien, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, David Warde-Farley, Ian Goodfellow, Arnaud Bergeron, et al. Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*, volume 3. Citeseer, 2011.
- [2] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.
- [3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

- [4] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo Moitinho de Almeida, Brian McFee, Hendrik Weideman, Gábor Takács, Peter de Rivaz, Jon Crall, Gregory Sanders, Kashif Rasul, Cong Liu, Geoffrey French, and Jonas Degraeve. Lasagne: First release., August 2015.
- [5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [6] Joe Lemley, Shabab Bazrafkan, and Peter Corcoran. Deep learning for consumer devices and services: Pushing the limits for machine learning, artificial intelligence, and computer vision. *IEEE Consumer Electronics Magazine*, 6(2):48–56, 2017.
- [7] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.
- [8] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- [9] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.
- [10] OpenCV. Opencv face detection using haar cascades, 2018.
- [11] Zhou Wang and Alan C Bovik. A universal image quality index. *IEEE signal processing letters*, 9(3):81–84, 2002.
- [12] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Appendices

A Diversity Measurements

1. **MSE (Mean Squared Error)**: MSE measures the average of the squares of the errors or deviations; representing the difference between the estimator and what is estimated. The lower value of MSE shows lesser error.

$$MSE(f, g) = \frac{1}{mn} \sum_0^{m-1} \sum_0^{n-1} \|f(i, j) - g(i, j)\| \quad (16)$$

2. **RMSE (Root Mean Squared Error)**: RMSE is a quadratic scoring rule that measures the average magnitude of the error. It is the square root of the average of squared differences between prediction and actual observation. The lower value of RMSE shows lesser error.

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (17)$$

3. **MAE (Mean Absolute Error)**: MAE also measures the average magnitude of the errors in a set of predictions, without considering their direction. It is the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight. The lower value of MAE shows lesser error.

$$MAE(f, y) = \frac{1}{n} \sum_{i=1}^n |f_i - y_i| \quad (18)$$

4. **UQI (Universal Quality Index)** [11]: UQI measures the structural distortion of the images by modeling the distortion as a combination of three factors: loss of correlation, luminance distortion, and contrast distortion. The higher value of UQI shows lesser error.
5. **SSIM (Structural Similarity Index)** [12]: SSIM is a perception-based model that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. The higher value of SSIM shows lesser error.

Appendix H

Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN), Multi Class Scenarios

Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN), Multi Class Scenarios

Training Conditional Generators

Shabab Bazrafkan · Peter Corcoran

Received: date / Accepted: date

Abstract Conditional generators learn the data distribution for each class in a multi-class scenario and generate samples for a specific class given the right input from the latent space. In this work, a method known as “Versatile Auxiliary Classifier with Generative Adversarial Network” for multi-class scenarios is presented. In this technique, the Generative Adversarial Networks (GAN)’s generator is turned into a conditional generator by placing a multi-class classifier in parallel with the discriminator network and backpropagate the classification error through the generator. This technique is versatile enough to be applied to any GAN implementation. The results on two databases and comparisons with other method are provided as well.

Keywords Conditional deep generators · Generative Adversarial Networks · Machine learning

1 Introduction

With emerge of affordable parallel processing hardware, it became almost impossible to find any aspect of Artificial Intelligence (AI) that Deep Learning (DL) has not been applied to [9]. DL provides superior outcomes on classification and regression problems compared to classical machine learning methods. The impact of DL is not limited to such problems, but also generative models are taking advantage of these techniques in learning

data distribution for big data scenarios where classical methods fail to provide a solution. Generative Adversarial Networks (GAN) [6] utilise Deep Neural Network capabilities and are able to estimate the data distribution for large size problems. These models comprise two networks, a generator, and a discriminator. The generator makes random samples from a latent space, and the discriminator determines whether the sample is adversarial, made by the generator, or is genuine image coming from the dataset. GANs are successful implementations of deep generative models, and there are multiple variations such as WGAN [1], EBGAN [15], BEGAN [3], ACGAN [11], and DCGAN [13], which have evolved from the original GAN by altering the loss function and/or the network architecture. Variational Autoencoders (VAE) [7] are the other successful implementation of deep generative models. In these models the bottleneck of a conventional autoencoder is considered as the latent space of the generator, i.e., the samples are fed to an autoencoder, and besides the conventional autoencoders loss function, the KullbackLeibler (KL) divergence between the distribution of the data at the bottleneck is minimized compared to a Gaussian distribution. In practice, this is achieved by adding the KL divergence term to the means square error of the autoencoder network. The biggest downside to VAE models is their blurry outputs due to the mean square error loss [5]. PixelRNN and PixelCNN [12] are other famous implementations of the deep neural generative models. PixelRNN is made of 2-dimensional LSTM units, and in PixelCNN, a Deep Convolutional Neural Network is utilized to estimate the distribution of the data. Training conditional generators are one of the most appealing applications of GAN. Conditional GAN (CGAN) [10] and Auxiliary Classifier GAN (ACGAN) [11] are among the most utilized schemes for this purpose. Wherein

S. Bazrafkan
Cognitive, Connected & Computational Imaging Research
National University of Ireland Galway
Tel.: +353-83-466-7835
E-mail: s.bazrafkan1@nuigalway.ie

P. Corcoran
Cognitive, Connected & Computational Imaging Research
National University of Ireland Galway

the CGAN approach uses the auxiliary class information alongside with partitioning the latent space and ACGAN improves the CGAN idea by introducing a classification loss which back-propagates through the discriminator and generator network. The CGAN method is versatile enough to apply to every variation of GAN. But ACGAN is restricted to a specific loss function which decreases its adaptivity to other GAN varieties. In [?], the ACGAN technique is extended to be applicable to any GAN implementation for binary problems (2 class scenarios). The technique is known as Versatile Auxiliary Classifier with Generative Adversarial Network (VAC+GAN) and is implemented by placing a classifier in parallel with the discriminator and back-propagate the classification error through the generator alongside the GAN's loss.

This work expand the original VAC+GAN [?] idea to multi-class scenarios. In this approach, the classifier is trained independently from the discriminator which gives the opportunity of applying it to any variation of GAN. The main contribution of VAC+GAN is its versatility, and proofs are provided to show the applicability of the method regardless of the GAN structure or loss functions.

In the next section the VAC+GAN for multi-class scenarios is explained. And in the third section the implementations of the ACGAN and VAC+GAN is presented alongside with the comparisons with other methods. The discussions and future works are given in the last section.

2 Versatile Auxiliary Classifier + Generative Adversarial Network (VAC+GAN)

The concept proposed in this research is to place a classifier network in parallel with the Discriminator. The classifier accepts the samples from the generator, and the classification error is back-propagated through the classifier and the generator. The model structure is shown in figure 1.

In this section it is shown that by placing a classifier at the output of the generator and minimizing the categorical cross-entropy as the classifiers loss, the Jensen-Shannon Divergence between all the classes is increased. The terms used in the mathematical proofs are as follows:

1. N is the number of the classes.
2. The latent space Z is partitioned in to $\{Z_1, Z_2, \dots, Z_N\}$ subsets. This means that $\{Z_1, Z_2, \dots, Z_N\}$ are disjoint and their union is equal to the Z -space.
3. C is the classifier function.
4. \mathcal{L}_{ce} is the binary cross-entropy loss function.

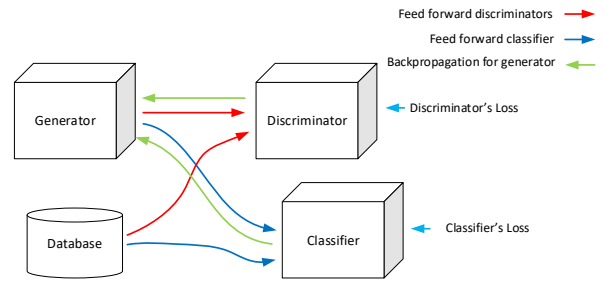


Fig. 1 The presented model for training conditional deep generators.

5. \mathcal{L}_{ce} is the categorical cross-entropy loss function.

Proposition 1. *In the multiple classes case, the classifier C has N outputs, where N is the number of the classes. In this approach, each output of the classifier corresponds to one class. For a fixed Generator and Discriminator, the optimal output for class c (c 'th output) is:*

$$C_{G,D}^*(c) = \frac{p_{X_c}(\mathbf{x})}{\sum_{i=1}^N p_{X_i}(\mathbf{x})} \quad (1)$$

Proof. Considering just one of the outputs of the classifier, the categorical cross-entropy can be reduced to binary cross-entropy given by

$$\begin{aligned} \mathcal{L}_{ce}(C(c)) = & -\mathbb{E}_{\mathbf{z} \sim p_{Z_c}(\mathbf{z})} [\log(C(G(\mathbf{z})))] \\ & - \mathbb{E}_{\mathbf{z} \sim \sum_{i \neq c} p_{Z_i}(\mathbf{z})} [1 - \log(C(G(\mathbf{z})))] \end{aligned} \quad (2)$$

which is equal to

$$\begin{aligned} \mathcal{L}_{ce}(C(c)) = & \int \left(p_{Z_c}(\mathbf{z}) \log(C(G(\mathbf{z}))) \right. \\ & \left. + \left(\sum_{i \neq c} p_{Z_i}(\mathbf{z}) \log(1 - C(G(\mathbf{z}))) \right) d\mathbf{z} \right) \end{aligned} \quad (3)$$

By considering $G(\mathbf{z}_i) = \mathbf{x}_i$ we have

$$\begin{aligned} \mathcal{L}_{ce}(C(c)) = & \int \left(p_{X_c}(\mathbf{x}) \log(C(\mathbf{x})) \right. \\ & \left. + \left(\sum_{i \neq c} p_{X_i}(\mathbf{x}) \log(1 - C(\mathbf{x})) \right) d\mathbf{x} \right) \end{aligned} \quad (4)$$

The function $f \rightarrow m \log(f) + n \log(1 - f)$ gets its maximum at $\frac{m}{m+n}$ for any $(m, n) \in \mathbb{R}^2 \setminus \{0, 0\}$, concluding the proof. \square

Theorem 1 *The maximum value for $\mathcal{L}_{ce}(C)$ is $N \log(N)$ and is achieved if and only if $p_{X_1} = p_{X_2} = \dots = p_{X_N}$.*

Proof. The categorical cross-entropy is given by

$$\begin{aligned}\mathcal{L}_{cce} &= - \sum_{i=1}^N \mathbb{E}_{\mathbf{z} \sim p_{Z_i}(\mathbf{z})} [\log(C(G(\mathbf{z})))] \\ &= - \sum_{i=1}^N \int p_{X_i}(\mathbf{x}) \log(C(\mathbf{x})) d\mathbf{x}\end{aligned}\quad (5)$$

From equation 1 we have

$$\begin{aligned}\mathcal{L}_{cce} &= - \sum_{i=1}^N \left(\int p_{X_i}(\mathbf{x}) \log \left(\frac{p_{X_i}(\mathbf{x})}{\sum_{j=1}^N p_j(\mathbf{x})} \right) d\mathbf{x} \right) \\ &= - \sum_{i=1}^N \left(\int p_{X_i}(\mathbf{x}) \log \left(\frac{p_{X_i}(\mathbf{x})}{\sum_{j=1}^N \frac{p_j(\mathbf{x})}{N}} \right) d\mathbf{x} \right) + N \log(N) \\ &= N \log(N) - \sum_{i=1}^N KL \left(p_{X_i}(\mathbf{x}) \left\| \sum_{j=1}^N \frac{p_{X_j}(\mathbf{x})}{N} \right. \right)\end{aligned}\quad (6)$$

Where KL is the Kullback-Leibler divergence, which is always positive or equal to zero.

Now consider $p_{X_1} = p_{X_2} = \dots = p_{X_N}$. From 6 we have

$$\begin{aligned}\mathcal{L}_{cce} &= N \log(N) - \sum_{i=1}^N KL \left(p_{X_i}(\mathbf{x}) \left\| p_{X_i}(\mathbf{x}) \right. \right) \\ &= N \log(N)\end{aligned}\quad (7)$$

concluding the proof. \square

Theorem 2 *Minimizing \mathcal{L}_{cce} increases the Jensen-Shannon Divergence between $p_{X_1}, p_{X_2}, \dots, p_{X_N}$*

Proof. From equation 6 we have

$$\begin{aligned}\mathcal{L}_{cce} &= N \log(N) \\ &- \int \sum_{i=1}^N \left(p_{X_i}(\mathbf{x}) \left[\log(p_{X_i}(\mathbf{x})) \right. \right. \\ &\quad \left. \left. - \log \left(\sum_{j=1}^N \frac{p_{X_j}(\mathbf{x})}{N} \right) \right] \right) d\mathbf{x}\end{aligned}\quad (8)$$

Which can be rewritten as

$$\begin{aligned}\mathcal{L}_{cce} &= N \log(N) - \sum_{i=1}^N \left(\int p_{X_i}(\mathbf{x}) \log(p_{X_i}(\mathbf{x})) d\mathbf{x} \right) \\ &+ \underbrace{\int \sum_{i=1}^N \left(p_{X_i}(\mathbf{x}) \log \left(\sum_{j=1}^N \frac{p_{X_j}(\mathbf{x})}{N} \right) \right) d\mathbf{x}}_{\left(\sum_{i=1}^N p_{X_i}(\mathbf{x}) \right) \left(\log \left(\sum_{j=1}^N \frac{p_{X_j}(\mathbf{x})}{N} \right) \right)}\end{aligned}\quad (9)$$

Which is equal to

$$\begin{aligned}\mathcal{L}_{cce} &= N \log(N) \\ &- N \sum_{i=1}^N \left(\frac{1}{N} \int p_{X_i}(\mathbf{x}) \log(p_{X_i}(\mathbf{x})) d\mathbf{x} \right) \\ &+ N \int \left(\sum_{i=1}^N \frac{p_{X_i}(\mathbf{x})}{N} \right) \left(\log \left(\sum_{j=1}^N \frac{p_{X_j}(\mathbf{x})}{N} \right) \right) d\mathbf{x}\end{aligned}\quad (10)$$

This equation can be rewritten as

$$\begin{aligned}\mathcal{L}_{cce} &= N \log(N) \\ &- \left[H \left(\sum_{i=1}^N \frac{1}{N} p_{X_i}(\mathbf{x}) \right) - \sum_{i=1}^N \frac{1}{N} H(p_{X_i}(\mathbf{x})) \right]\end{aligned}\quad (11)$$

wherein the $H(p)$ is the Shannon entropy of the distribution p .

The Jensen Shannon divergence between N distributions p_1, p_2, \dots, p_N , is defined as

$$\begin{aligned}JSD_{\pi_1, \pi_2, \dots, \pi_N} (p_1, p_2, \dots, p_N) &= H \left(\sum_{i=1}^N \pi_i p_i \right) \\ &- \sum_{i=1}^N \pi_i H(p_i)\end{aligned}\quad (12)$$

From equations 11 and 12 we have

$$\begin{aligned}\mathcal{L}_{cce} &= N \log(N) \\ &- N JSD_{\frac{1}{N}, \frac{1}{N}, \dots, \frac{1}{N}} (p_{X_1}(\mathbf{x}), p_{X_2}(\mathbf{x}), \dots, p_{X_N}(\mathbf{x}))\end{aligned}\quad (13)$$

Minimizing \mathcal{L}_{cce} is increasing the JSD term, concluding the proof. \square

In this section it has been shown that by placing a classifier at the output of the generator and back-propagate the classification error throughout the generator one can increase the dis-similarity between the classes for generator and therefore train a deep generator that can produce class specified samples. In the next section the proposed idea is implemented for multi-class cases and also compared with state of the art methods.

3 Experimental Results

In this section, two main experiments are explained to show the effectiveness of VAC+GAN. The first one is on MNIST database and visual comparisons with CGAN, CDCGAN and ACGAN is presented. The second experiment is on CFAR10 dataset and the classification error is compared against ACGAN method. All the networks are trained in Lasagne [4] on top of Theano [2] library in Python, unless stated otherwise.

Table 1 the generator structure for the MNIST+DCGAN experiment. All deconvolution layers are using (2,2) padding with stride (2,2).

Layer	Type	kernel	Activation
Input	Input(10)	–	–
Hidden 1	Dense	1024	ReLU
BatchNorm 1	–	–	–
Hidden 2	Dense	$128 \times 7 \times 7$	ReLU
BathNorm 2	–	–	–
Hidden 3	Deconv	5×5 (64ch)	ReLU
BathNorm 3	–	–	–
Output	Deconv	5×5 (1ch)	Sigmoid

Table 2 the discriminator structure for the MNIST+DCGAN experiment. All convolution layers are using (2,2) padding with stride (2,2).

Layer	Type	kernel	Activation
Input	Input	–	–
Hidden 1	Conv	5×5 (64 ch)	LeakyR(0.2)
BatchNorm 1	–	–	–
Hidden 2	Conv	5×5 (128 ch)	LeakyR(0.2)
BathNorm 2	–	–	–
Hidden 3	Dense	1024	LeakyR(0.2)
Output	Dense	1	Sigmoid

Table 3 the classifier structure for the MNIST+DCGAN experiment.

Layer	Type	Kernel	Activation
Input	Input(28×28)	–	–
Hidden 1	Conv	3×3 (16 ch)	ReLU
Pool 1	Max pooling	2×2	–
Hidden 2	Conv	3×3 (8 ch)	ReLU
Pool 2	Max pooling	2×2	–
Hidden 3	Dense	1024	ReLU
Output	Dense	10	Softmax

3.1 MNIST

In this experiment, the performance of the proposed method is investigated on MNIST database. MNIST (“Modified National Institute of Standards and Technology”) is known as the “hello world” dataset of computer vision. It is a historically significant image classification benchmark introduced in 1999, and there has been a considerable amount of research published on MNIST image classification. MNIST contains 60,000 training images and 10,000 test images, both drawn from the same distribution. It consists of 28×28 pixel images of handwritten digits. Each image is assigned a single truth label digit from $[0, 9]$.

The proposed method has been applied to the DCGAN scheme. The Generator, Discriminator and the Classifier used in this experiment are given in tables 1, 2 and 3 respectively. And the loss function for the proposed

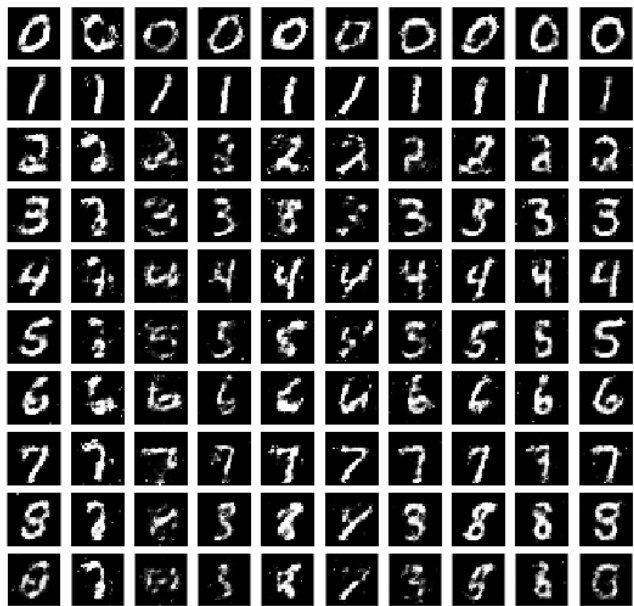


Fig. 2 Samples drawn from conditional generator trained using CGAN scheme on MNIST dataset. each row corresponds to one class.

method (VAC+GAN) is given by:

$$\begin{aligned} L_g &= \vartheta \cdot BCE(G(z|c), 1) + \zeta \cdot CCE \\ L_d &= BCE(x, 1) + BCE(G(z|c), 0) \end{aligned} \quad (14)$$

where, L_g , and L_d are the generator and discriminator losses respectively, G is the generator function, BCE is the binary cross-entropy loss for discriminator and CCE is the categorical cross-entropy loss for the classifier. In this experiment, ϑ and ζ are equal to 0.2 and 0.8 respectively.

The optimizer used for training the generator and discriminator is ADAM with learning rate, β_1 and β_2 equal to 0.0002, 0.5 and 0.999 respectively. And the classifier is optimized using nestrov momentum gradient descent with learning rate and momentum equal to 0.01 and 0.9 respectively. The results of the conditional generators trained using Conditional GAN (CGAN)¹, Conditional DCGAN (CDCGAN)², ACGAN³, and proposed method (VAC+GAN) on MNIST dataset are shown in figures 2, 3, 4⁴, and 5 respectively.

As it is shown in these figures the presented method gives superior results compare to CGAN and CDCGAN while using the exact same structure of generator as in CDCGAN. The results are comparable with ACGAN

¹ <https://github.com/znxlwm/tensorflow-MNIST-cGAN-cDCGAN>

² <https://github.com/znxlwm/tensorflow-MNIST-cGAN-cDCGAN>

³ <https://github.com/buriburisuri/ac-gan>

⁴ <https://github.com/buriburisuri/ac-gan/blob/master/png/sample.png>

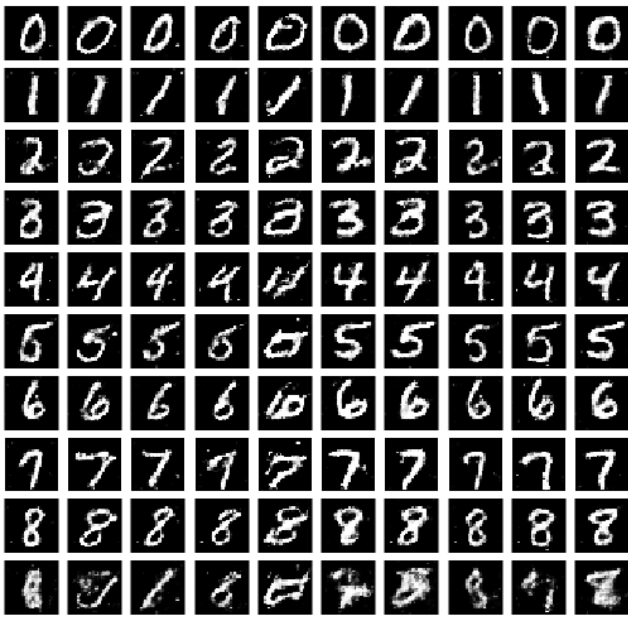


Fig. 3 Samples drawn from conditional generator trained using CDCGAN scheme on MNIST dataset. each row corresponds to one class.

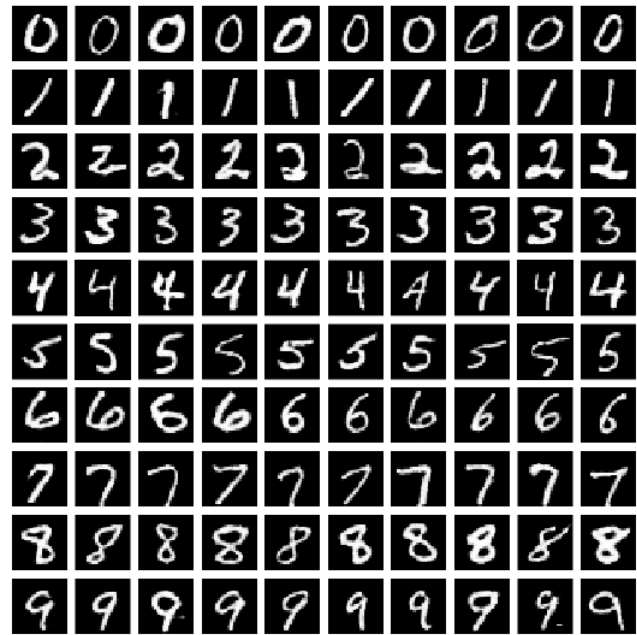


Fig. 5 Samples drawn from conditional generator trained using proposed scheme (VAC+GAN) on MNIST dataset. each row corresponds to one class.

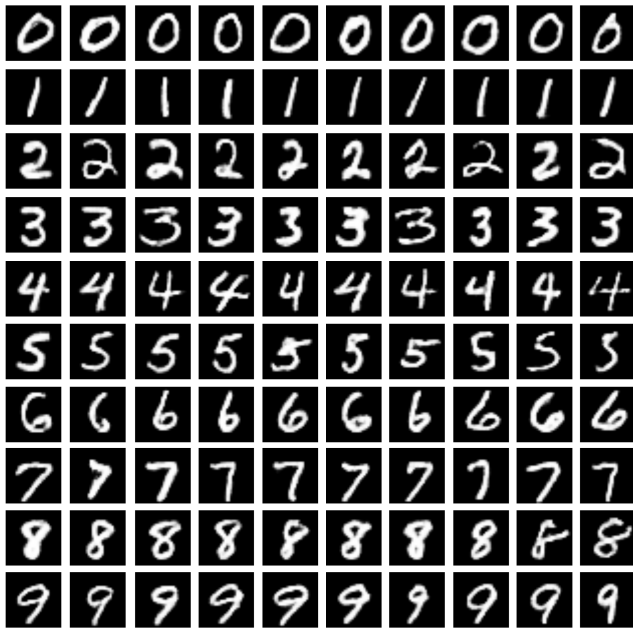


Fig. 4 Samples drawn from conditional generator trained using ACGAN scheme on MNIST dataset. each row corresponds to one class.

and the difference here is that this method is more versatile and can be applied to any GAN model regardless of model architecture and loss function.

Table 4 the generator structure for the CFAR10 experiment. All deconvolution layers are using 'SAME' padding with stride (2,2).

Layer	Type	kernel	Activation
Input	Input	-	-
Hidden 1	Dense	384×4	ReLU
Reshape	Reshape	$384 \text{ch } 4 \times 4$	-
Hidden 2	DeConv	5×5 (192 ch)	ReLU
BathNorm 2	-	-	-
Hidden 3	DeConv	5×5 (96 ch)	ReLU
BathNorm 3	-	-	-
Output	DeConv	5×5 (3 ch)	tanh

3.2 CFAR10

The CFAR10 database [8] consists of 60000 images in 10 classes wherein 50000 of these images are for training and 10000 for testing purposes. The next experiment is comparing ACGAN⁵ to VAC+GAN method on generating images and also the classification accuracy of these methods are compared. Networks utilized in this experiment are shown in tables 4,5 and 6 correspond to generator, discriminator⁶ and classifier respectively. The same generator and discriminator architectures have been used in both implementations to obtain fair comparisons.

⁵ <https://github.com/King-Of-Knights/Keras-ACGAN-CIFAR10>

⁶ <https://github.com/King-Of-Knights/Keras-ACGAN-CIFAR10/blob/master/cifar10.py>

Table 5 the discriminator structure for the CFAR10 experiment. All deconvolution layers are using 'SAME' padding with kernel size 3×3 , *st* stands for stride size and MBDisc is Mini Batch Discrimination layer explained in [14].

Layer	Type	kernel	Activation
Input	Input	$32 \times 32 \times 3$	–
Gaussian	Noise	$\sigma = 0.05$	–
Hidden 1	Conv	16ch <i>st</i> (2, 2)	LeakyR(0.2)
DropOut 1	DropOut	$p = 0.5$	–
Hidden 2	Conv	32ch <i>st</i> (1, 1)	LeakyR(0.2)
BathNorm 1	–	–	–
DropOut 2	DropOut	$p = 0.5$	–
Hidden 3	Conv	64ch <i>st</i> (2, 2)	LeakyR(0.2)
BathNorm 2	–	–	–
DropOut 3	DropOut	$p = 0.5$	–
Hidden 4	Conv	128ch <i>st</i> (1, 1)	LeakyR(0.2)
BathNorm 3	–	–	–
DropOut 4	DropOut	$p = 0.5$	–
Hidden 5	Conv	256ch <i>st</i> (2, 2)	LeakyR(0.2)
BathNorm 4	–	–	–
DropOut 5	DropOut	$p = 0.5$	–
Hidden 6	Conv	512ch <i>st</i> (1, 1)	LeakyR(0.2)
BathNorm 5	–	–	–
DropOut 6	DropOut	$p = 0.5$	–
MBDisc [14]	–	–	–
Output	Dense	1	sigmoid

Table 6 the classifier structure for the CFAR10 experiment.

Layer	Type	kernel	Activation
Input	Input	$32 \times 32 \times 3$	–
Hidden 1	Conv	5×5 (128ch)	ReLU
BatchNorm 1	–	–	–
MaxPool 1	MaxPool	(2,2)	–
Hidden 2	Conv	5×5 (256ch)	ReLU
BatchNorm 2	–	–	–
MaxPool 2	MaxPool	(2,2)	–
Hidden 3	Conv	5×5 (512ch)	ReLU
BatchNorm 3	–	–	–
MaxPool 3	MaxPool	(2,2)	–
Hidden 4	Dense	512	ReLU
Output	Dense	10	softmax

The loss function used to train the VAC+GAN is given by

$$\begin{aligned} L_g &= \vartheta \cdot BCE(G(z|c), 1) + \zeta \cdot CCE \\ L_g &= BCE(x, 1) + BCE(G(z|c), 0) \end{aligned} \quad (15)$$

where, L_g , and L_d are the generator and discriminator losses respectively, G is the generator function, BCE is the binary cross-entropy loss for discriminator and CCE is the categorical cross-entropy loss for the classifier. In this experiment, ϑ and ζ are equal to 0.5 and 0.5 respectively.

The optimizer used for training the generator and discriminator is ADAM with learning rate, β_1 and β_2 equal to 0.0002, 0.5 and 0.999 respectively. And the classifier is optimized using nestrov momentum gradient descent with learning rate and momentum equal to 0.01 and

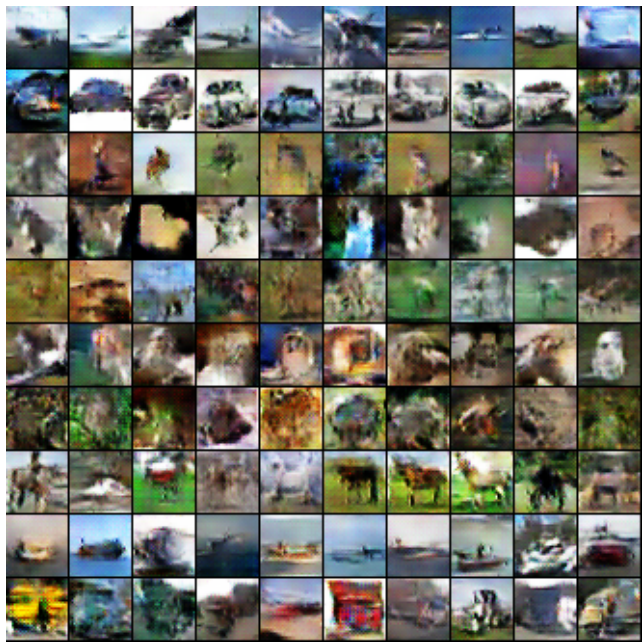


Fig. 6 Generated samples using ACGAN.

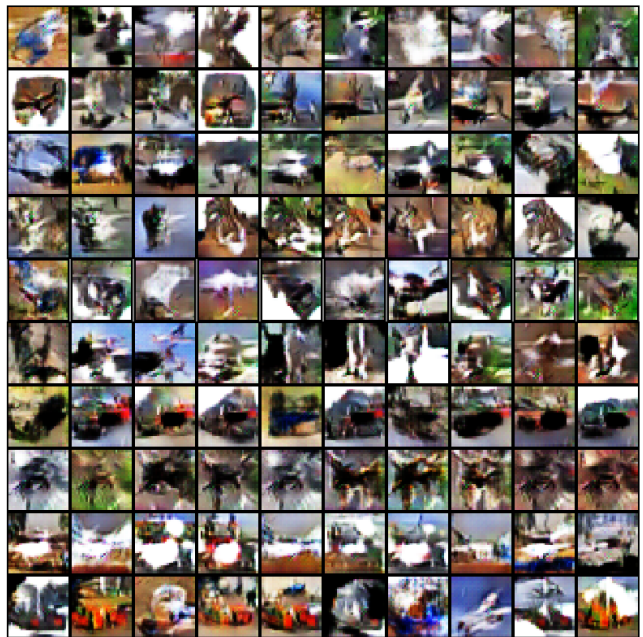


Fig. 7 Generated samples using VAC+GAN.

0.9 respectively. The results for ACGAN and proposed method are shown in figures 6⁷ and 7 respectively. The CFAR10 database is an extremely unconstrained and there are just 10000 samples in each class. Therefore the output of both implementations are vague and in order to compare these methods the classification errors are compared. The confusion matrix for ACGAN and

⁷ https://github.com/King-Of-Knights/Keras-ACGAN-CIFAR10/blob/master/plot_epoch_220_generated.png

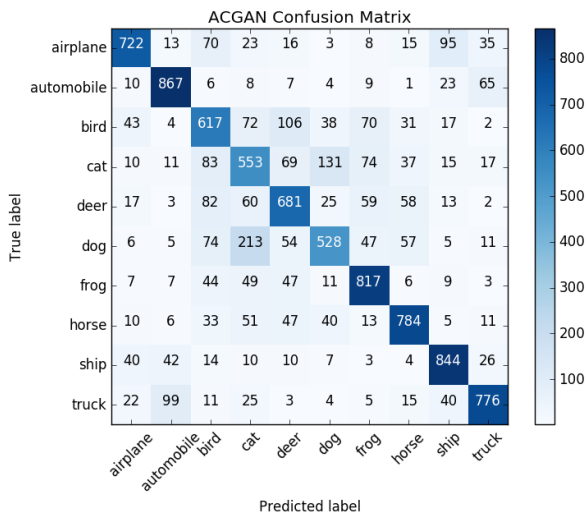


Fig. 8 Confusion matrix for ACGAN method on CFAR10.

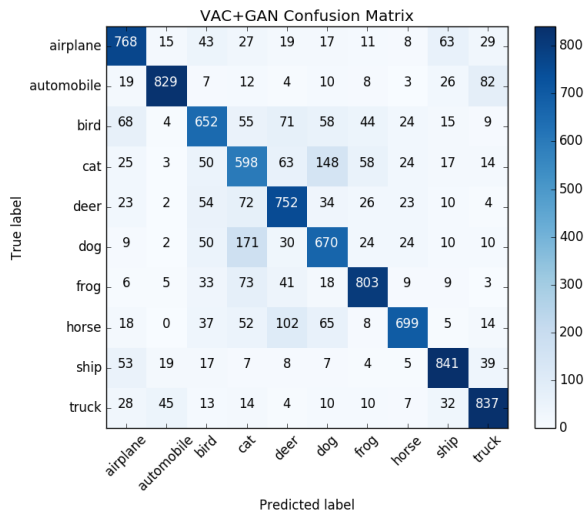


Fig. 9 Confusion matrix for VAC+GAN method on CFAR10.

VAC+GAN are shown in figures 8⁸ and 9 respectively.

Confusion matrices show the better classification performed by the VAC+GAN compared to the ACGAN. Classification accuracies for ACGAN and VAC+GAN on CFAR10 are 71.89% and 74.49% respectively after 200 epochs. The proposed method gives higher accuracy. The main advantage of the proposed method is the versatility in choosing the proper classifier network while in the ACGAN method the classification task is restrained to discriminator because the discriminator is performing as classifier as well. The VAC+GAN method is versatile in choosing the GAN scheme as well. It can be applied to any GAN implementation just by placing a classifier in parallel with discriminator.

⁸ https://github.com/King-Of-Knights/Keras-ACGAN-CIFAR10/blob/master/Confusion_Matrix.png

4 Discussion and Conclusion

In this work, a new approach introduced to train conditional deep generators. It also has been proven that VAC+GAN is applicable to any GAN framework regardless of the model structure and/or loss function (see Sec 2) for multi class problems. The idea is to place a classifier in parallel to the discriminator network and back-propagate the classification loss through the generator network in the training stage.

It has also been shown that the presented framework increases the Jensen Shannon Divergence (JSD) between classes generated by the deep generator. i.e., the generator can produce more distinct samples for different classes which is desirable.

The results has been compared to the implementation of CGAN, CDCGAN and ACGAN on MNIST dataset and also the comparisons are given on CFAR10 dataset with respect to ACGAN method. The ACGAN gives comparable results, but the main advantage of the proposed method is its versatility in choosing the GAN scheme and also the classifier architecture.

The future work includes applying the method to datasets with larger number of classes and also extend the implementation for bigger size images. The other idea is to apply this method to regression problems

Acknowledgements This research is funded under the SFI Strategic Partnership Program by Science Foundation Ireland (SFI) and FotoNation Ltd. Project ID: 13/SPP/I2868 on Next Generation Imaging for Smartphone and Embedded Platforms.

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: International Conference on Machine Learning, pp. 214–223 (2017)
2. Bergstra, J., Bastien, F., Breuleux, O., Lamblin, P., Pascanu, R., Delalleau, O., Desjardins, G., Warde-Farley, D., Goodfellow, I., Bergeron, A., et al.: Theano: Deep learning on gpus with python. In: NIPS 2011, BigLearning Workshop, Granada, Spain, vol. 3. Citeseer (2011)
3. Berthelot, D., Schumm, T., Metz, L.: Began: Boundary equilibrium generative adversarial networks. arXiv preprint arXiv:1703.10717 (2017)
4. Dieleman, S., Schlter, J., Raffel, C., Olson, E., Snderby, S.K., Nouri, D., Maturana, D., Thoma, M., Battenberg, E., Kelly, J., Fawc, J.D., Heilman, M., de Almeida, D.M., McFee, B., Weideman, H., Takcs, G., de Rivaz, P., Crall, J., Sanders, G., Rasul, K., Liu, C., French, G., Degraeve, J.: Lasagne: First release. (2015). DOI 10.5281/zenodo.27878. URL <http://dx.doi.org/10.5281/zenodo.27878>
5. Frans, K.: Variational autoencoders explained (2016). URL <http://kvfrans.com/variational-autoencoders-explained/>
6. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.:

- Generative adversarial nets. In: *Advances in neural information processing systems*, pp. 2672–2680 (2014)
7. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013)
 8. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
 9. Lemley, J., Bazrafkan, S., Corcoran, P.: Deep learning for consumer devices and services: Pushing the limits for machine learning, artificial intelligence, and computer vision. *IEEE Consumer Electronics Magazine* **6**(2), 48–56 (2017)
 10. Mirza, M., Osindero, S.: Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014)
 11. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585* (2016)
 12. Oord, A.v.d., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759* (2016)
 13. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015)
 14. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: *Advances in Neural Information Processing Systems*, pp. 2234–2242 (2016)
 15. Zhao, J., Mathieu, M., LeCun, Y.: Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126* (2016)

Appendix I

Versatile Auxiliary Regressor with Generative Adversarial network (VAR+GAN)

Versatile Auxiliary Regressor with Generative Adversarial network (VAR+GAN)

Shabab Bazrafkan, Peter Corcoran
National University of Ireland Galway

Abstract

Being able to generate constrained samples is one of the most appealing applications of the deep generators. Conditional generators are one of the successful implementations of such models wherein the created samples are constrained to a specific class. In this work, the application of these networks is extended to regression problems wherein the conditional generator is restrained to any continuous aspect of the data. A new loss function is presented for the regression network and also implementations for generating faces with any particular set of landmarks is provided.

Keywords: Generative Adversarial Networks, Conditional Generators, Face Generation

1. Introduction

Generative Adversarial Networks (GAN) [1] are among the most successful implementations of deep generators. The idea of GAN is to train two agents, a generator, and a discriminator, simultaneously. The generator is a deep neural network which accepts a vector from a latent space (uniformly distributed noise) and outputs a sample, same type of the database. The discriminator is a binary classifier determining whether this sample is generated or is a genuine data coming from the database. The training is accomplished by playing a min-max game between these two networks. There are several extensions to the original GAN idea wherein the original GAN is adapted to a specific condition by changing the network structures and/or loss function. For example, Conditional GAN (CGAN) [2], Auxiliary Classifier GAN (ACGAN) [3], and Versatile Auxiliary Classifier with GAN (VAC+GAN) [4] are utilizing the original GAN idea to train conditional generators wherein the

output of the generator is constrained to a specific class given the right input sequence. CGAN does this by partitioning the latent space and also the auxiliary knowledge of the data class. In ACGAN the loss of the CGAN is manipulated by adding a classification term which back-propagates through generator and discriminator. The VAC+GAN extends the ACGAN scheme to be more adaptable to different GAN variations. This is done by adding a classifier network in parallel with the discriminator network, and the classification error is back-propagated through the generator.

In this work, the idea of VAC+GAN is extended to regression problems by replacing the classifier with a regression network. A new loss function is presented for this network. The regression error is back-propagated through the generator. This gives the opportunity to train a generator while constraining the generated samples to any continuous aspect of the original database.

Similar ideas include the scheme presented in [5] wherein a Hierarchical Generative Model (HGM) is utilized for eye image synthesis and eye gaze estimation. This work introduces a variation of GAN known as conditional Bidirectional GAN (cBiGAN) which is a mixture of CGAN and Bidirectional GAN (BiGAN). The main issue with this method is the lack of adaptability to every GAN variation. This method is only applicable to BiGAN scheme. This approach is shown in figure 1. In our observations, cBiGAN implementation is able to generate samples for each aspect but there are very low variations in generated samples for a specific aspect. The proposed VAR+GAN scheme produces higher variations in the same condition. This is discussed in more details in section 3.4. The other advantage of the proposed method is its versatility, i.e., it applies to any GAN implementation regardless of the network architecture and/or loss function.

In the next section, the idea of VAR+GAN is explained alongside with the presented loss function for regression network. Section 3 explains implementation, results and the comparisons for the presented method against cBiGAN and the conclusions and future works is discussed in the last section.

2. Versatile Auxiliary Regressor with Generative Adversarial network (VAR+GAN)

The idea of proposed scheme is to place a regression network in parallel with the discriminator and back-propagate the regression error through the generator (see figure 2). In this method, the generator is constrained to generate samples with specific continuous aspects. For example, in the face

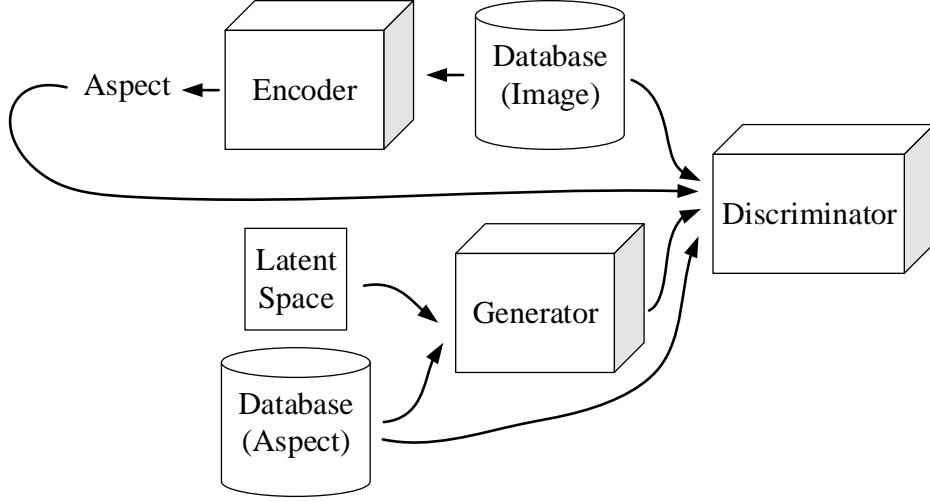


Figure 1: cBiGAN scheme used for training conditional generators.

generation application, given the right latent sequence, the generator creates faces with particular landmarks. The following loss function is introduced for the regression network

$$L_R = \int \int dp_z(\mathbf{z}) \left(-\log \left(1 - (y - R(G(\mathbf{z}))) \right) \right) dz \quad (1)$$

wherein \mathbf{z} is the latent space variable, $dp_z(\mathbf{z})$ is the distribution of an infinitesimal partition of latent space, y is the target variable (ground truth), R is the regression function and G is the generator function.

Proposition 1. For the loss function in equation 1 the optimal regressor is

$$R^* = \frac{p(\mathbf{x})}{c} + y - 1 \quad (2)$$

wherein $p(\mathbf{x})$ is the distribution of the generator's output, c is post-integration constant, and y is the target function.

Proof. Considering the inner integration of equation 1 and by replacing $G(\mathbf{z}) = \mathbf{x}$, the extremum of the loss function with respect to R is

$$\frac{d}{dR} \int dp_x(\mathbf{x}) \left(-\log \left(1 - (y - R(\mathbf{x})) \right) \right) dx = 0 \quad (3)$$

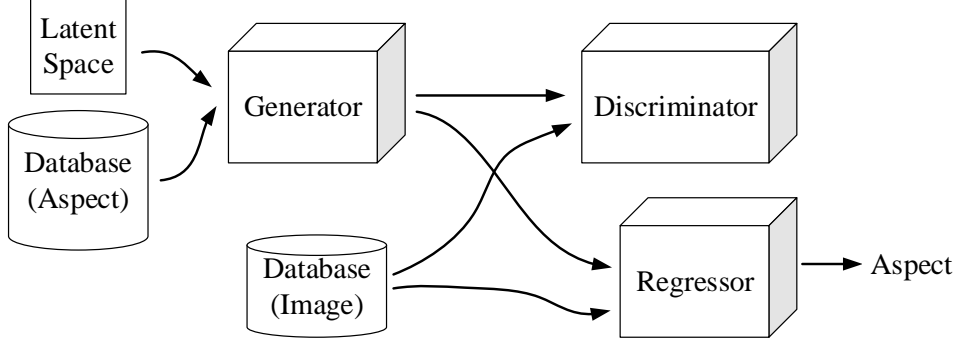


Figure 2: Presented method (VAR+GAN) for training conditional generators.

which can be written as

$$\int \frac{-dp_x}{R - y + 1} = 0 \quad \Rightarrow \quad \frac{p_x}{R - y + 1} = c \quad (4)$$

this results in

$$R = \frac{p(\mathbf{x})}{c} + y - 1 \quad (5)$$

concluding the proof. \square

Theorem 1. *Minimizing the loss function in equation 1 decreases the entropy of the generator's output.*

Proof. by replacing equation 2 in 1 we have

$$L_R = \int \int -\log\left(\frac{p_x(\mathbf{x})}{c}\right) dp_x dx \quad (6)$$

which can be rewritten as

$$L_R = - \int p_x(\mathbf{x}) \log(p_x(\mathbf{x})) dx + \log(c) = H(p_x(\mathbf{x})) + \log(c) \quad (7)$$

wherein H is the shannon entropy. Minimizing L_R results in decreasing $H(p_x(\mathbf{x}))$ concluding the proof. \square

Adding the regressor to the model decreases the entropy of the generated samples. This is expectable since the idea is to constrain the output of the generator to obey some particular criteria. This is shown in observations in section 3.4.

Theorem 2. For any two sets of samples and their corresponding targets (y_1 and y_2), the loss function in equation 1 increases the Jensen Shannon Divergence (JSD) between generated samples for these two sets.

Proof. Consider \mathbf{z}_1 and \mathbf{z}_2 are two partitions of the latent space correspond to two sets of samples with targets y_1 and y_2 . In this case, the loss function in equation 1 is given by:

$$L_R = - \int p_{z_1}(\mathbf{z}) \log(1 - (y_1 - R(G(\mathbf{z}_1)))) dz - \int p_{z_2}(\mathbf{z}) \log(1 - (y_2 - R(G(\mathbf{z}_2)))) dz \quad (8)$$

Considering $G(\mathbf{z}_1) = \mathbf{x}_1$, $G(\mathbf{z}_2) = \mathbf{x}_2$, $c_1 = 1 - y_1$, and $c_2 = 1 - y_2$ equation 8 simplifies to

$$L_R = - \int p_{x_1}(\mathbf{x}) \log(c_1 + R(\mathbf{x})) dx - \int p_{x_2}(\mathbf{x}) \log(c_2 + R(\mathbf{x})) dx \quad (9)$$

To find the optimum $R(x)$ the derivative of the integrand is set to zero given by

$$\frac{p_{x_1}}{c_1 + R} + \frac{p_{x_2}}{c_2 + R} = 0 \quad (10)$$

which results in

$$R = - \frac{p_{x_1} c_2 + p_{x_2} c_1}{p_{x_1} + p_{x_2}} \quad (11)$$

By replacing equation 11 in equation 9 it simplifies to

$$L_R = - \int p_{x_1} \log \left(\frac{(c_1 - c_2)p_{x_1}}{p_{x_1} + p_{x_2}} \right) + \int p_{x_2} \log \left(\frac{(c_2 - c_1)p_{x_2}}{p_{x_1} + p_{x_2}} \right) dx \quad (12)$$

which can be rewritten as

$$R_L = - \int \log \left(\frac{p_{x_1}}{\frac{p_{x_1} + p_{x_2}}{2}} \right) - \int \log \left(\frac{p_{x_2}}{\frac{p_{x_1} + p_{x_2}}{2}} \right) - \log(c_1 - c_2) - \log(c_2 - c_1) - \log(4) \quad (13)$$

which equals to

$$R_L = - \log(c_1 - c_2) - \log(c_2 - c_1) - \log(4) - 2JSD(p_{x_1} || p_{x_2}) \quad (14)$$

minimizing R_L increasing $JSD(p_{x_1} || p_{x_2})$ term, concluding the proof. \square

In this section, it has been shown that the presented loss function increases the distance between generated samples for any two set of aspects, which is desirable.

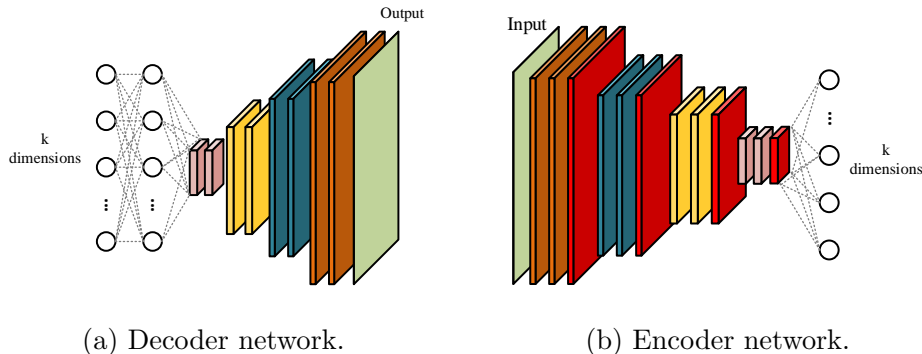


Figure 3: Network architectures used for implementation purposes.

3. Implementation and Results

In this section, the implementation of the VAR+GAN is presented and compared to cBiGAN method. To keep the consistency in comparisons, the same architecture for generator network has been kept throughout all implementations. All the networks are trained using Lasagne [6] library on the top of Theano [7] library in Python.

3.1. Network Architectures

Three main architectures used in this section are encoder, decoder, and regression networks. The first two are shown in figure 3. The decoder contains one fully connected layer which maps the input to a 3D layer. Next layers are all convolutional layers followed by (2, 2) un-pooling layers for every second convolution. The exponential linear unit (ELU) [8] is used as activation function except in the last layer wherein no non-linearity has been applied except for the encoder in cBiGAN scheme wherein tanh nonlinearity is applied in the output layer. The encoder network is made of convolutional layers with ELU activation function. The downscaling in these layers is obtained by using (2, 2) stride in every second convolutional layer. In the decoder network, all convolutional layers have 64 channels while in the encoder, the number of the channels is gradually increased to 128, 192, and 256 after each pooling layer. The layers shown in red are applying no nonlinearity to their input. The regression network is a conventional deep neural network shown in table 1.

Table 1: the Regression network used in experiments.

Layer	Type	kernel	Activation
Input	Input(48 × 48)	–	–
Hidden 1	Conv	3 × 3(64 ch)	ReLU
Pool 1	Max pooling	2 × 2	–
Hidden 2	Conv	3 × 3(64 ch)	ReLU
Pool 2	Max pooling	2 × 2	–
Hidden 3	Dense	1024	ReLU
Output	Dense	98	Tanh

3.2. Database

The dataset used in this work is CelebA database [9] which is made of 202,599 frontal posed images. Face regions are cropped and resized to 48 × 48 pixels using OpenCV frontal face cascade classifier [10]. Supervised Descent Method (SDM) [11] is used for facial point detection. The detector is based on [12] and it utilizes the discriminative 3D facial deformable model to find 49 facial landmarks including contours of eyebrows, eyes, mouth and the nose. These landmarks are used as the data aspect in this work.

3.3. Implementation

3.3.1. VAR+GAN

For the proposed scheme (figure 2), the Boundary Equilibrium Generative Adversarial Network (BEGAN) [13] is utilized to train the deep generator. In this method the generator architecture is same as the decoder network shown in figure 3a with input dimension $k = 128$. The discriminator is an auto-encoder network wherein the decoder architecture is same as the generator and the encoder is the network shown in figure 3b with $k = 128$. The regression network is shown in table 1 and the loss function for proposed implementation is a modified version of the original BEGAN loss [13] given by:

$$\begin{aligned}
 L_d &= L(x) - k_t \cdot L(G(z|y)) \\
 L_g &= \vartheta \cdot L(G(z|y)) + \zeta \cdot L_R \\
 k_{t+1} &= k_t + \lambda_k (\gamma L(x) - L(G(z|y)))
 \end{aligned}
 \tag{15}$$

Where L_g and L_d are generators and discriminators losses respectively. G is the generator function, z is a sample from the latent space, x and y are

genuine image and corresponding ground truth drawn from the database , λ_k is the learning rate for k , γ is the equilibrium hyper parameter set to 0.5 in this work, L_R is the regression loss given by equation 1, and ϑ and ζ are set to 0.97 and 0.03 respectively, and L is the auto-encoders loss defined by

$$L(v) = |v - D(v)|^2 \quad (16)$$

The optimizer used for training the generator and discriminator is ADAM with learning rate, β_1 and β_2 equal to 0.0001, 0.5 and 0.999 respectively. And the regression network is optimized using nestrov momentum gradient descent with learning rate and momentum equal to 0.01 and 0.9 respectively.

3.3.2. cBiGAN

The cBiGAN scheme is implemented for the same generator architecture and on the same database to make fair comparisons with the proposed method. The generator architecture is same as the decoder network shown in figure 3a with input dimension $k = 128$. The discriminator model is same as the encoder network in figure 3b with $k = 1$ and *sigmoid* non-linearity at the output layer. And the encoder network in figure 1 has the architecture shown in figure 3b with $k = 98$ and *tanh* non-linearity at the output layer. The loss function for this scheme is presented in [5] given by

$$\begin{aligned} L_D &= \log(p^r) + \log(1 - p^I) + \log(1 - p^s) \\ L_G &= \log(p^I) \\ L_E &= \log(p^s) + \theta ||s^- - y||^2 \end{aligned} \quad (17)$$

wherein s^- is the encoder's output, y is the genuine aspect coming from the database, and

$$p^r = D(y, x) \quad , \quad p^I = D(y, G(z|y)) \quad , \quad p^s = D(s^-, x) \quad (18)$$

where D and G are discriminator and generator functions respectively, x and y are genuine image and corresponding ground truth drawn from the database, and z is a sample from the latent space. The coefficient θ is set to 0.8. The optimizer used for training the model is ADAM with learning rate, β_1 and β_2 equal to 0.0001, 0.5 and 0.999 respectively.



(a) Proposed method (VAR+GAN).

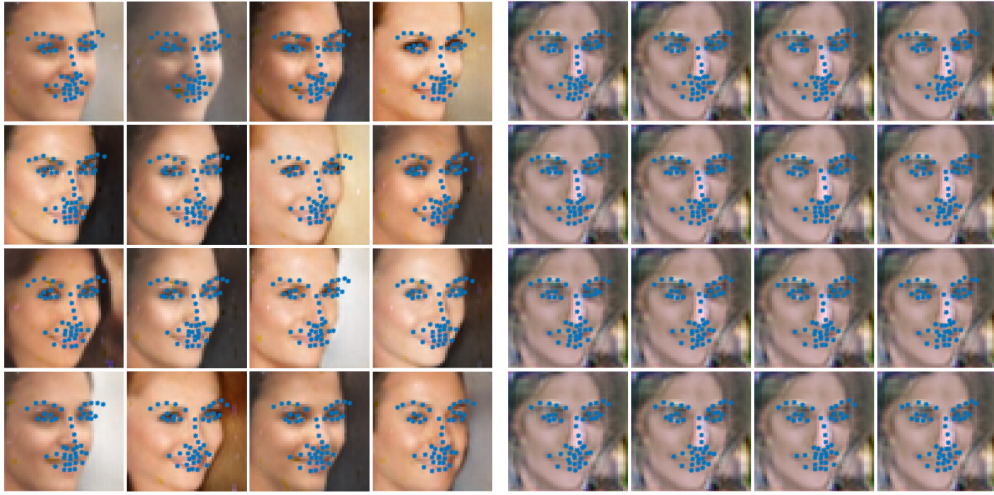
(b) cBiGAN.

Figure 4: Generator outputs for proposed method (VAR+GAN) vs cBiGAN.

3.4. Results

In this section, the proposed method is compared against the cBiGAN method while generating faces for a particular landmark point set. The results for six sets of landmarks are shown in figures 4 to 9. In each figure (left) the outputs from the proposed method for a particular set of landmarks are illustrated while in right side of the figures the output of the generator trained in cBiGAN scheme is given for the same landmarks.

As shown in these figures, both methods are able to generate samples constrained to a particular set of landmarks but the proposed method generates higher variations of faces for a given landmark set while cBiGAN fails to create different samples in the same condition. The advantage of VAR+GAN is the versatility of the method which facilitates the implementation and also guarantees the higher quality in generated samples. For example in this work the proposed method is taking advantage of simplicity and power of BEGAN implementation and only change applied is to place a regression network and add its error value to the generator’s loss. While cBiGAN method is constrained to a specific loss function which is a big disadvantage.



(a) Proposed method (VAR+GAN).

(b) cBiGAN.

Figure 5: Generator outputs for proposed method (VAR+GAN) vs cBiGAN.



(a) Proposed method (VAR+GAN).

(b) cBiGAN.

Figure 6: Generator outputs for proposed method (VAR+GAN) vs cBiGAN given particular landmarks.



(a) Proposed method (VAR+GAN).

(b) cBiGAN.

Figure 7: Generator outputs for proposed method (VAR+GAN) vs cBiGAN given particular landmarks.



(a) Proposed method (VAR+GAN).

(b) cBiGAN.

Figure 8: Generator outputs for proposed method (VAR+GAN) vs cBiGAN given particular landmarks.



(a) Proposed method (VAR+GAN).

(b) cBiGAN.

Figure 9: Generator outputs for proposed method (VAR+GAN) vs cBiGAN given particular landmarks.

4. Conclusion and Future Work

In this work, a new scheme for training conditional deep generators has been introduced wherein the generator is able to constrain the generated samples to any continuous aspect of the dataset. The presented method is versatile enough to be applicable to any GAN variation with any network structure and loss function. The idea is to place a regression network in parallel with the discriminator network and back-propagate the regression error through the generator. A new loss function is also presented and it has been shown that it increases the JSD between data generated for any two set of aspects. The other property of the proposed loss is the reduction of the entropy for generated samples which is expectable because of the constraints applied to generated data.

The proposed method is also compared with the only available method with the same purpose (to the best of our knowledge in the time writing the article). cBiGAN method generates samples with a particular aspect but there are very low variations between generated samples while VAR+GAN produces higher variations for a specific set of aspects. Being able to generate variable samples is crucial for tasks including augmentation purposes. Being able to augment the database for certain data aspects and use them in

training the final products is one of the most interesting applications for the conditional generators.

The future works include merging the VAC+GAN and VAR+GAN methods to constrain the generator to create samples from a specific class with a particular continuous aspect. And also investigating the influence of the generated samples in augmentation task for different applications.

References

References

- [1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [2] M. Mirza, S. Osindero, Conditional generative adversarial nets, arXiv preprint arXiv:1411.1784.
- [3] A. Odena, C. Olah, J. Shlens, Conditional image synthesis with auxiliary classifier gans, arXiv preprint arXiv:1610.09585.
- [4] S. Bazrafkan, H. Javidnia, P. Corcoran, Versatile auxiliary classifier+generative adversarial network (vac+ gan); training conditional generators, arXiv preprint arXiv:1805.00316.
- [5] K. Wang, R. Zhao, Q. Ji, A hierarchical generative model for eye image synthesis and eye gaze estimation.
- [6] S. Dieleman, J. Schlter, C. Raffel, E. Olson, S. K. Snderby, D. Nouri, D. Maturana, M. Thoma, E. Battenberg, J. Kelly, J. D. Fauw, M. Heilman, D. M. de Almeida, B. McFee, H. Weideman, G. Takcs, P. de Rivaz, J. Crall, G. Sanders, K. Rasul, C. Liu, G. French, J. Degraeve, Lasagne: First release. (Aug. 2015). doi:10.5281/zenodo.27878.
URL <http://dx.doi.org/10.5281/zenodo.27878>
- [7] J. Bergstra, F. Bastien, O. Breuleux, P. Lamblin, R. Pascanu, O. Delalleau, G. Desjardins, D. Warde-Farley, I. Goodfellow, A. Bergeron, et al., Theano: Deep learning on gpus with python, in: *NIPS 2011, BigLearning Workshop*, Granada, Spain, Vol. 3, Citeseer, 2011.

- [8] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), arXiv preprint arXiv:1511.07289.
- [9] Z. Liu, P. Luo, X. Wang, X. Tang, Deep learning face attributes in the wild, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3730–3738.
- [10] D. Cristinacce, T. F. Cootes, Feature detection and tracking with constrained local models., in: Bmvc, Vol. 1, 2006, p. 3.
- [11] X. Xiong, F. De la Torre, Supervised descent method and its applications to face alignment, in: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE, 2013, pp. 532–539.
- [12] A. Asthana, S. Zafeiriou, S. Cheng, M. Pantic, Incremental face alignment in the wild, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 1859–1866.
- [13] D. Berthelot, T. Schumm, L. Metz, Began: Boundary equilibrium generative adversarial networks, arXiv preprint arXiv:1703.10717.