



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

Title	Pushing the AI envelope: merging deep networks to accelerate edge artificial intelligence in consumer electronics devices and systems
Author(s)	Bazrafkan, Shabab; Corcoran, Peter
Publication Date	2018-02-08
Publication Information	Bazrafkan, S., & Corcoran, P. M. (2018). 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. doi: 10.1109/MCE.2017.2775245
Publisher	IEEE
Link to publisher's version	https://dx.doi.org/10.1109/MCE.2017.2775245
Item record	http://hdl.handle.net/10379/14583
DOI	http://dx.doi.org/10.1109/MCE.2017.2775245

Downloaded 2024-05-24T03:44:58Z

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



Merging Deep Neural Networks to Accelerate Edge AI in CE Devices & Systems

By: Shabab Bazrafkan and Peter Corcoran

Introduction:

Deep Neural Networks (DNN) are being used widely by both academic and industry researchers to solve many long-standing problems in machine learning. In fact there has been such a growth of research in this field and it has been applied to so many varying problems that it would not be untrue to say that we may be living through the pre-cursor of the singularity [1].

But regardless of your views on Artificial Intelligence there can be 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 long-standing problems in computer vision for autonomous vehicles. And many of these problems can have very useful applications in the design of smarter consumer electronic systems and devices.

Now the question for CE engineers is how to leverage this wealth of academic and industry research efforts, turning them into practical DNN solutions that are suitable for deployment in practical devices and electronic systems?

Finding the Forest

There are a number of challenges here. Firstly, academic research is typically focused on a specific aspect of a research problem so the authors can compare their results with different approaches to the same problem, showing significant improvement in certain aspects at the expense of reduced performance, accuracy or robustness. Rarely does a new technique or approach perform so well that all previous research methods become obsolete. For the engineer approaching this problem with a broader perspective he 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 a lot of resources are thrown at the problem in hand and even where optimizations are investigated – and, thankfully, the deep learning research community has become more sensitive to their importance – these optimizations apply to a particular DNN architecture. In general researchers are focused on designing a network to solve the immediate problem at hand 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 to their problem, each adopting different network structures and optimizations while still sharing many common networks layers and data inputs. However little of the literature is concerned with techniques for merging and optimizing a combination of existing DNN approaches into a more holistic merged solution. In effect everyone is focused on growing their own cluster of trees and developing these as quickly as possible, rather than working with others to develop and manage a forest where everyone could take advantage of economies of scale.

Fortunately, most DNN architectures do share many common elements, just like trees in a forest. And it is possible to bring them 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 consumer electronics deployments. And that is what we will now focus on in the remainder of this article.

A Crash Course in Deep Networks

Here we give a quick refresher of some of the most important elements of a DNN. For CE magazine readers who are unfamiliar with deep learning, our previous article [2] is recommended reading.

Now 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, fully connected layers, followed by pooling, and regularization layers. Data flows through these layers in sequence

eventually emerging in a modified data structure that typically enables a binary interpretation on 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 leading to the descriptive term ‘deep’ to be applied to such networks.

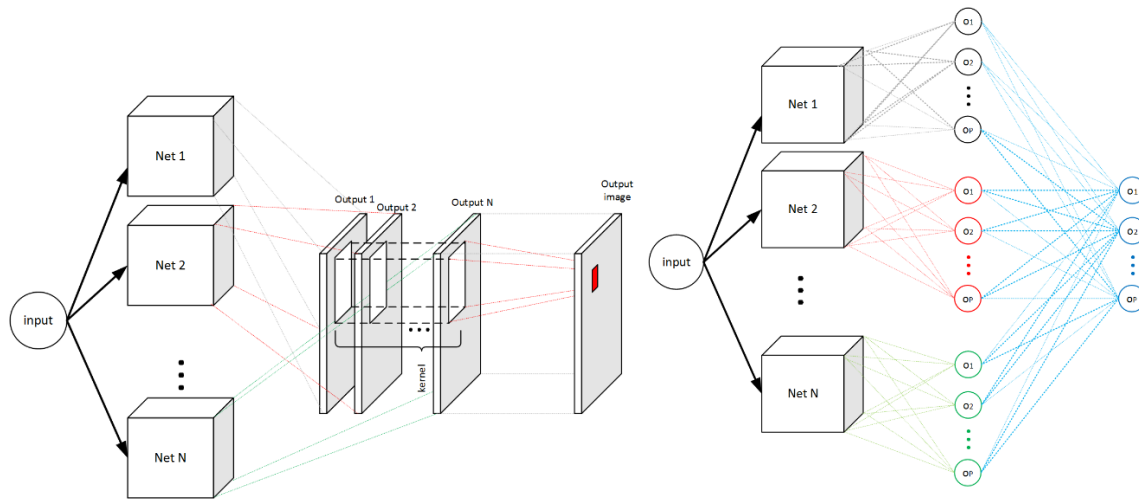


Figure 1: concatenating the last layer into a single layer (convolution case)

Figure 2: concatenating the last layer into a single layer (Fully connected case)

The Main Categories of Deep Network Layer

Deep networks comprise a succession of data processing layers, each layer taking inputs from a preceding layer and filtering a set of inputs to generate a set of outputs. The best known of these layers are **convolutional layers** which 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:

$$P = I * W + b$$

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 types of layer are also used in deep network architectures. **Pooling layers** typically apply a non-linear transform on the input data with is normally used to reduce the size of the input data. They can be thought of as data concentrator layers and are important to control the overall size of a network which could grow too large if only convolutional layers were used.

A **fully-connected layer** is exactly the same as the classical Neural Network (NN) layers where all the neurons in a layer are connected to all the neurons in the subsequent layer. The neurons give the summation of their input multiplied by their weights and 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.

A **regularization layer** is used to prevent the overfitting inside the network. Different kind of regularizations have been proposed, the most important ones being (i) weight regularization, (ii) drop-out regularization where some data are skipped, and (iii) batch normalization where output data is averaged across several input data. Each of these regularization techniques has advantages and drawbacks which make them more (or less) suitable for specific applications.

The Costs of Going Deep

As was 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. So much so, that faced with almost any contemporary machine learning problem, it is almost inevitable that one can find a plethora 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 some particular 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 it may be desirable to use several different networks that can provide complimentary 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 additional compute cycles to process the added layers or a 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 the resource requirements quickly lead to impractical, even unfeasible solutions.

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. It would be nice if it was 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 the interested reader can find more detailed guidelines of the applications of these methods to 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 N different successful networks in the literature which each of them gives “reasonable” results on that specific task. Each of these networks has their own drawbacks and would fail on some input data.

As discussed above it would not make sense to implement multiple parallel networks due to the large resource requirements. It is known that deep networks often share some common layers near the output or input ends of the network, but it will still be necessary to duplicate the bulk of each network in order 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 these parallel networks into a final output from the DNN. See figures 1 and 2.

Now it would be nice 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 datasets. And that, dear reader, is what we are next going to outline to you.

Let’s Try an Example ...

To understand our methodology it is best to provide a practical working example.

In fact the original of this methodology were somewhat accidental and the results were somewhat unexpected – when this was first applied the expectation was that the new architecture derived from several individual DNNs would simply behave as a vote-taker.

The Iris Segmentation Problem

As an example problem 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 iris biometrics to smartphones [8]. A key problem for correct operation of the biometric authentication chain on mobile devices is that of iris segmentation [9]–[11]. To address the challenges of this specific problem a large database of poor quality iris images was derived from the extended CASIA 1000 dataset [12]. The database is augmented 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 online¹. Then three existing iris segmentation models were trained on the modified dataset.

¹ https://github.com/C3Imaging/Deep-Learning-Techniques/tree/Iris_SegNet/DBaugmentation

1. The first network is an 8 layer fully convolutional deep neural network designed to the iris segmentation task. All layers are using 7x7 kernels. The Batch normalization technique is used after each convolutional layer to avoid overfitting and faster convergence.(Figure 3a).

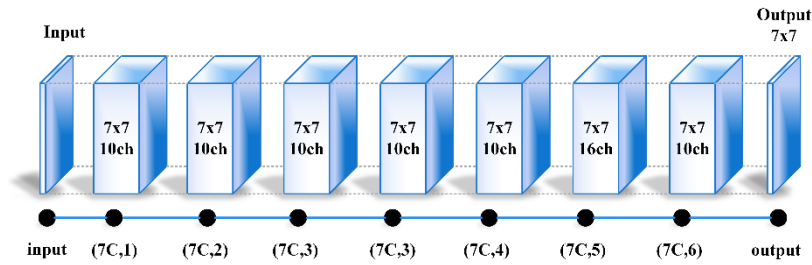


Figure 3a: network 1 used to perform iris segmentation.

2. The second network designed for the problem in hand is a 6 layer fully convolutional network shown in Figure 3b. The kernel size 3x3 has been used in all layers. No pooling is used in the network, and the batch normalization technique is used again after each convolutional layer.

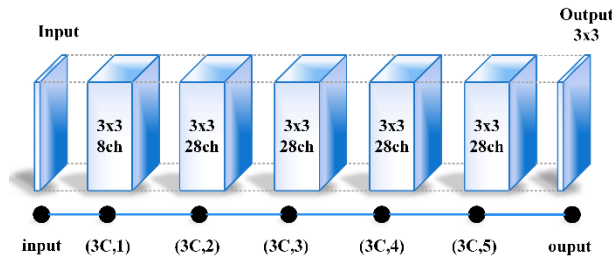


Figure 3b: network 2 used to perform iris segmentation.

3. The third proposed network is a 5 layer fully convolutional neural network shown in Figure 3c. The kernel size is increasing for each layer starting with 3x3 for the first layer and 11x11 for the output layer. No pooling is used in the network, and the batch normalization technique is used again after each convolutional layer.

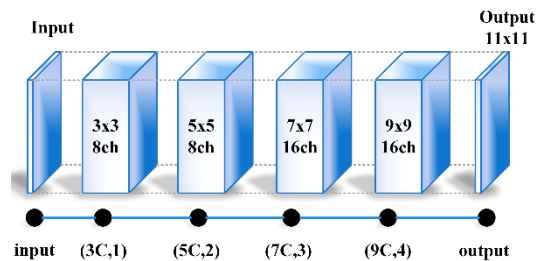


Figure 3c: network 3 used to perform iris segmentation.

Three networks described above are trained using Nesterov Momentum method for Binary Cross-Entropy loss function on expanded CASIA 1000 dataset.

Merging the Networks

The main purpose of the SPDNN model is to mix and merge several deep architectures and produce a final model which takes advantage of the specialised layers of each architecture, but is significantly smaller in size than the combined sizes of these networks. The approach adopted is based on graph optimisation.

- 1- Translate each network into its corresponding graph. (The graph representation for each network is shown at the bottom in Figs 3(a,b,c));
- 2- Assign properties to each node:
 - a. A 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; the interested reader is referred to [3] for

more complex examples). For example, 5C correspond to a convolutional layer with the 5x5 kernel.

- b. A second property is the distance of this layer to the input layer. For example, (9C,4) is the label of the node operating 9x9 convolutional operation and has distance 4 from the input data node.
- 3- In this step all these graphs are placed in parallel, sharing the same input and output. See Fig 4. All nodes with identical properties are assigned the same label. In this simple example, all nodes with the property (7C, 3) are assigned label C.

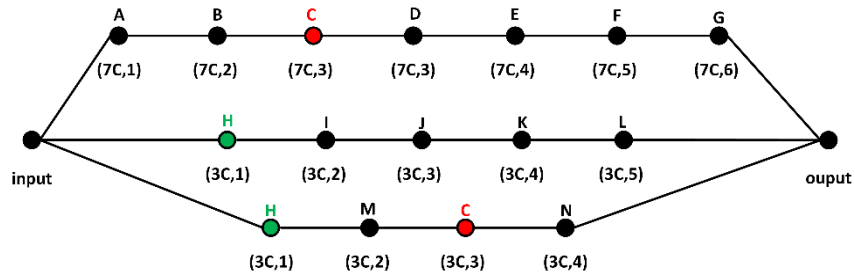


Figure 4: All graphs are placed in parallel sharing a single input and output

- 4- The graph contraction operation is applied to this graph. In graph contraction step all the nodes with the same label are merged into a single node while saving their connections to the previous and next nodes. See Fig 5.

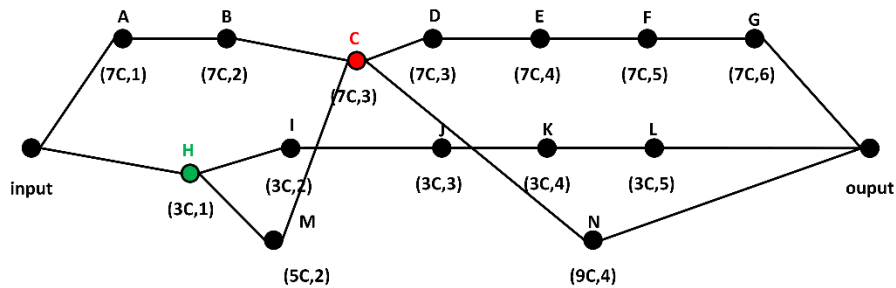


Figure 5: The graph contraction is applied to the graph in Figure

- 5- The last step is to translate the graph back into the neural network. In the nodes where two or more nodes are merging the concatenation operation is applied. And the operation for each node is specified by the first property of the node. The final network is shown in Figure

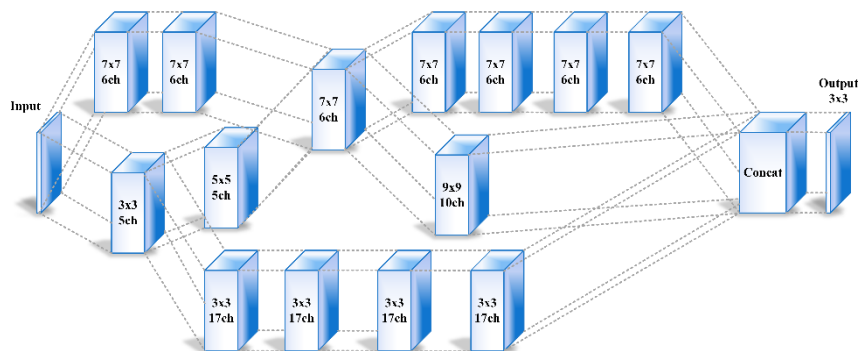


Figure 6: the contracted graph is translated into the neural network.

In order for the comparisons to be fair, the number of the channels in the final design is selected in a way for the network to have almost same number of parameters as each of its parent networks. This merged network model is now re-trained (Nestrov Momentum) and for the same loss function (Binary Cross-entropy). The same datasets used to training the original parent networks are utilised in this re-training procedure. Note that in this simplified example a single dataset derived from CASIA is employed, but in more complex problems the merged network responds well to cross-training with heterogeneous datasets.

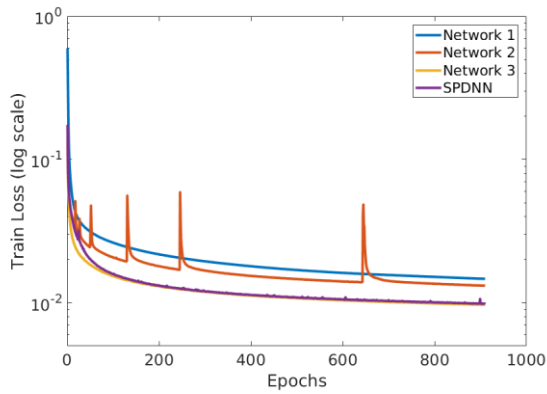


Figure 7a: Training loss for all networks including the merged (SPDNN) network.

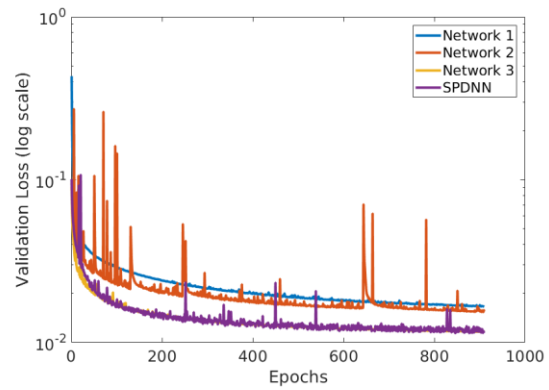


Figure 7b: Validation loss for all networks including the merged (SPDNN) network.

As you can see the losses converge to the best loss of the three networks. This shows that while having several networks using SPDNN helps the model to converge to the best available network.

The comparisons on the test dataset for all networks are shown in Fig 8 and Fig 9.

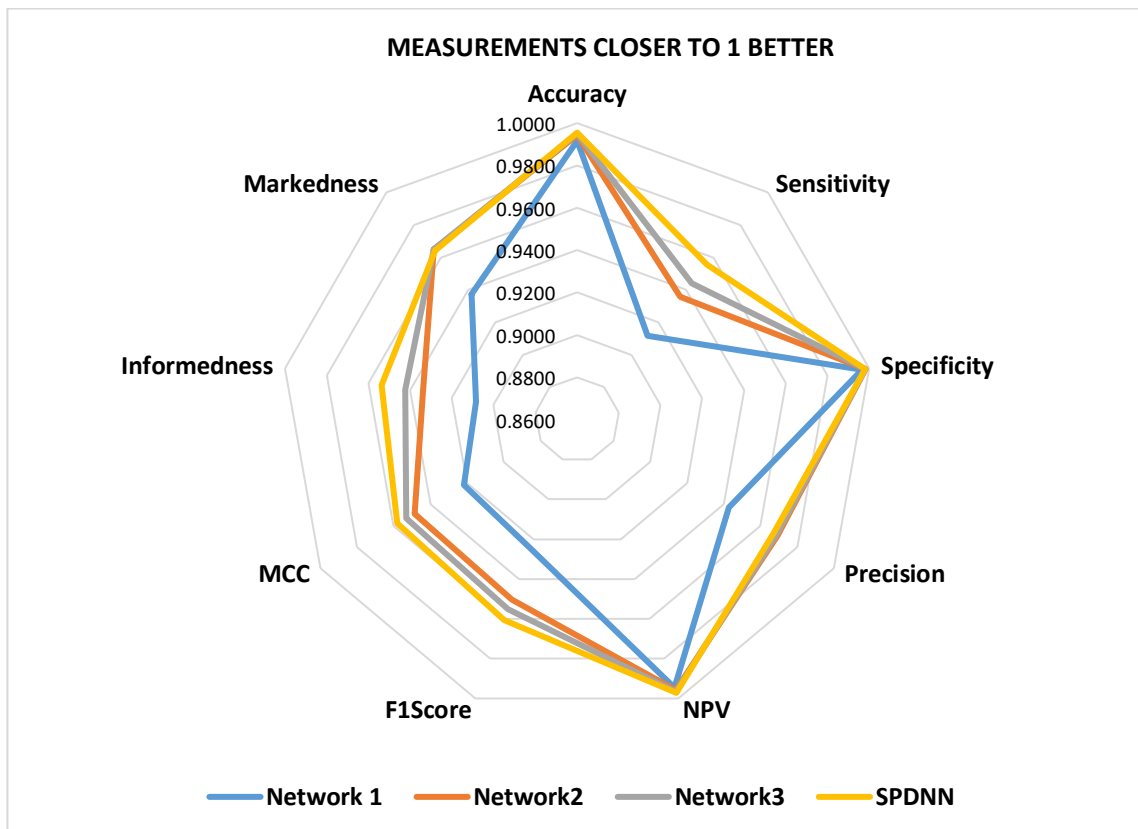


Figure 8: Comparisons for all networks. Higher values indicate better performance.

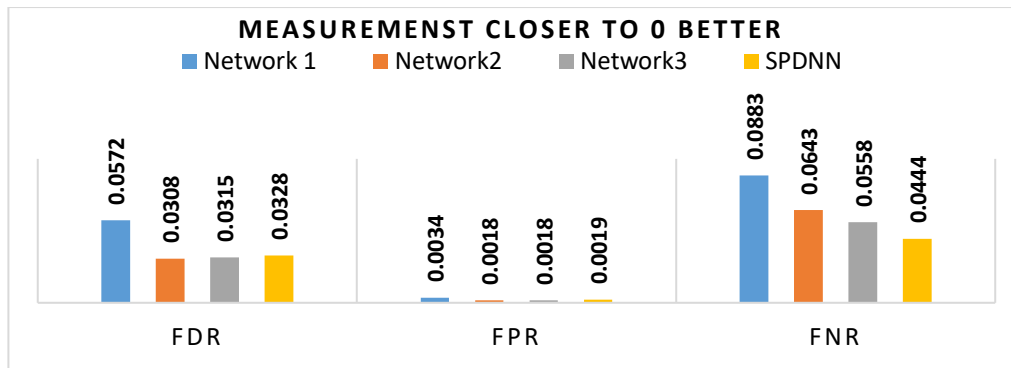


Figure 9: Comparisons for all networks. Lower values indicate better performance.

From Figs 8 & 9 it is concluded that, although the SPDNN network has the same number of parameters as the original networks 1, 2 or 3, it gives better results in most metrics including accuracy, sensitivity, NPV, F1Score, MCC, Informedness and FNR. In particular the higher accuracy, F1-score, and MCC show that the SPDNN idea has a better performance, in general, compared to its parent networks. Not all metrics are improved over the best of the 3 networks, but this is a relatively simple application of the methodology.

From the training and testing results we can see that our methodology helps the merged network to achieve an optimal design and increases the convergence in the training stage and in the test part. In more extensive studies on heterogenous datasets it has been shown that this methodology can (A) generalize the merged network beyond a trivial combination of parent networks, and (B) achieve significant reduction in size & computational scale for the merged network [3].

In plain English the merged network can be significantly smaller, often of a similar size to the largest of the individual parent networks, and performs 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 to apply these techniques to your own deep learning problems the authors encourage you to contact us and explore the potential for collaboration.

Acknowledgment

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).

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, "Depth from Monocular Images using a Semi-Parallel Deep Neural Network (SPDNN) Hybrid Architecture," *arXiv*, pp. 1–15, 2017.
- [4] S. Thavalengal and P. Corcoran, "Iris recognition on consumer devices — Challenges and progress," in *2015 IEEE International Symposium on 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 *Computer Vision and Pattern Recognition Workshops (CVPRW), 2015 IEEE Conference on*, 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*. 2013.
- [10] J. R. Matey, R. Broussard, and L. Kennell, "Iris image segmentation and sub-optimal images," *Image Vis. Comput.*, vol. 28, pp. 215–222, 2010.
- [11] S. Thavalengal, P. Bigioi, and P. Corcoran, "Efficient Segmentation for Multi-frame Iris Acquisition on Smartphones," in *2016 IEEE International Conference on Consumer Electronics (ICCE) (2016 ICCE)*, 2016, pp. 202–203.
- [12] "CASIA Iris Image Database." [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 *IEEE International Conference on Consumer Electronics, (ICCE 2018)*, 2018.

About The Authors

	<p>Shabab Bazrafkan</p> <p>Shabab Bazrafkan received his B.Sc degree from Urmia University, Urmia, Iran in electrical engineering in 2011 and M.Sc degree from Shiraz University of Technology (SuTECH) in telecommunication engineering, Image processing branch in 2013. Currently he is a PhD student at the Center for Cognitive, Connected & Computational Imaging, College of Engineering & Informatics, National University of Ireland, Galway (NUIG) and also works with Fotonation Ltd. His field of working is Deep Neural Networks and Neural Network design.</p> <p>Email: s.bazrafkan1@nuigalway.ie</p>
	<p>Prof. Peter Corcoran</p> <p>IEEE Fellow; 500+ technical publications & patents; 80+ peer reviewed papers & articles; 100+ international conference papers; co-inventor on 300+ granted US patents, University professor for 28 years; member IEEE Consumer Electronics Society 20+ years; Editor-in-Chief & Founding Editor of IEEE Consumer Electronics Magazine; former Vice-Dean of Research & Graduate Studies (7 year tenure) in the College of Engineering & Informatics at NUI Galway; co-founder of several start-up companies including FotoNation (www.fotonation.com); industry consultant & expert witness.</p> <p>Email: peter.corcoran@nuigalway.ie</p>