DOI: 10.2352/ISSN.2470-1173.2016.19.COIMG-149

Adaptive Activation Functions for Deep Networks

Michael Dushkoff, Raymond Ptucha; Rochester Institute of Technology; Rochester, NY/USA

Abstract

Artificial neural networks loosely mimic the complex web of nearly 100 trillion connections in the human brain. Deep neural networks, and specifically convolutional neural networks, have recently demonstrated breakthrough performances in the pattern recognition community. Studies on the network depth, regularization, filters, choice of activation function, and training parameters are numerous. With regard to activation functions, the rectified linear unit, is favored over the sigmoid and tanh function because the differentiation of larger signals is maintained. This paper introduces multiple activation functions per single neuron. Libraries have been generated to allow individual neurons within a neural network the ability to select between a multitude of activation functions, where the selection of each function is done on a node by node basis to minimize classification error. Each node is able to use more than one activation function if the final classification error can be reduced. The resulting networks have been trained on several commonly used datasets, which show increases in classification performance, and are compared to the recent findings in neuroscience research.

Introduction

The human brain has a network of billions of computational units, called neurons. The weighted sum of simultaneous inputs to a neuron determines its output spiking frequency which is subsequently passed on to other neurons. Each neuron is connected with up 10,000 other neurons, creating a network of 100 trillion synapses. Artificial neural networks (ANNs) loosely mimic a simplified version of this biological network of connections digitally and have been typically implemented in software, but recently in hardware as well [15]. Hierarchical arrangements of neurons into layers generally offer the most efficient ANNs [6]. The number of nodes in each layer, regularization strategies, addition of recurrent and skip forward connections, and network topologies have been studied in great detail [9]. To enable ANNs to learn complex nonlinear behaviors, each node applies an activation function to its output before passing it on to the next neuron. This traditionally monotonic function restricts the output range of the neuron using a sigmoidal or tanh function[5]. Many other activation functions have been studied, but the rectified linear units (ReLUs) [11], which clamp the negative outputs to zero and let the positive outputs go unchecked, have been the most successful as of late [4].

Recent neuroscience research indicates that biological neurons modify their activation functions as part of the learning process [13][14], some of which can be band pass oriented [3]. For smaller networks, studies on activation functions have shown that periodic activation functions allow ANNs to learn with fewer epochs and with fewer neurons [16][18][17][12][8]. These nonmonotonic functions allow more complex behavior in each layer, but can introduce chaotic tendencies during training if not regulated properly[7]. Prior studies have incorporated the activation function properties as parameters that are solved along with weights during backpropagation [2][16][10][17]. These studies have been done on small one to three layer networks which generally inflate the need for more complex activation functions which compensate for the simple networks with fewer neurons.

This research analyses the benefits of complex activation functions on larger deep networks. Each neuron in the deep network is configured to allow any number of activation functions, whereby the turning on and off of each activation function is learned during the training process. A new deep learning library built in the Torch7 framework incorporates the newly introduced activation functions and learns the weight parameters automatically during training. This new library allows nodes in a network to use a family of activation functions simultaneously in an effort to minimize classification error.

Background

Activation Functions

In models of the human brain, the activation functions within each neuron transfer the sum of all incoming synapse to an expected firing rate [1]. These activation functions can be symmetric or antisymmetric as they exhibit excitatory or inhibitory functions in the brain. Whenever a neuron becomes active, the concentration of ions on the cell's surface will change as well as the concentration of ions within the cell [13].

When neural networks were first gaining presence in the scientific community, Kurt Hornik proved mathematically that any feed-forward network with a single hidden layer containing a finite number of neurons could approximate any continuous function [19]. This theorem showed that neural networks are inherently universal approximators and that the activation itself does not give it this property, but rather the inherent structures of a feedforward network architecture. This theorem, however does not take into account the number of hidden units that would be required to approximate any function, nor does it take into account the feasibility of training such a network. From these implications, it is certainly important to consider the impact that the choice an activation function has on a specific network architecture's training time, and memory requirements on today's hardware.

Rectified linear units have been gaining popularity especially in deep convolutional neural network architectures. This nonsaturating nonlinear function trains faster than saturating nonlinear functions [11][20]. By allowing the positive values of the output of a neuron to grow unbounded, the resulting output of each convolutional layer is allowed to exhibit intensity equivariance [11] such that scaling intensities within regions of the image will not change the overall decision capabilities of the network. These qualities are important in classification problems that depend on inputs being invariant

DOI: 10.2352/ISSN.2470-1173.2016.19.COIMG-149

Adaptive Methods

There have been many attempts at creating adaptive activation functions in the past including adaptive cubic spline activation functions [16] as well as single function per layer optimization of rectified linear units [23]. Neither of these methods include more complex non-linear functions such as the sigmoid or hyperbolic tangent functions nor have either of these methods shown their effectiveness on modern datasets and neural network architectures. These pre-existing versions of adaptive functions have shown that there are clear advantages to providing adaptive capabilities for such functions such as reduced training times and increased accuracies. For these reasons, an adaptive model that can be easily integrated with existing techniques to improve overall accuracies of artificial neural networks has been explored to improve upon these concepts.

Adaptive Activation Functions

Optimization for each activation function was achieved by defining a convex cost function for the linear weighted combination of each activation function applied to an input. In order to better understand the operation that is being applied, this process can be visualized as a single entity in a neural network referred to as an "adaptive activation function layer" as shown in Figure 1. The individual activation functions are defined as f_i, where $i \in 1 \ldots N$. The function l refers to a continuous differentiable function which is applied to each gate matrix g_i. This gate matrix represents a matrix of parameters to be optimized to find the best activation function by gating certain activation functions, while allowing others to express themselves. In the case of a convolutional neural network, a node would be representative of a single pixel from one layer to the next which essentially allows for each pixel to have a separate activation function which is dynamically optimized.

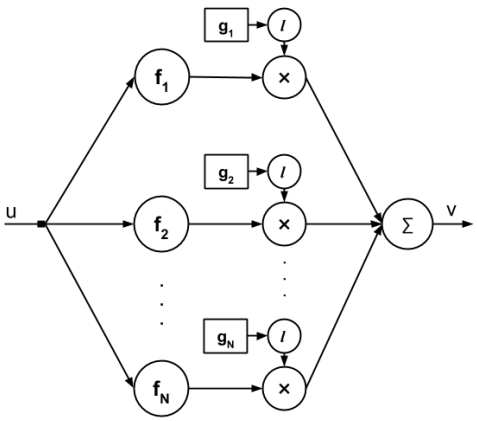

Figure 1: Block diagram of a single adaptive activation function layer.

Ideally, the values of $l(g_i)$ should fall within 0 to 1 which corresponds to having each activation function as either on or off respectively. These gate parameters are treated as optimization parameters in the gradient descent algorithm in order to allow for their optimal values to be solved for. The output v is calculated by passing the input u through this layer as defined by (1).

$$v = \sum_{i=1}^{N} f_i(u) l(g_i) \tag{1}$$

This adaptive layer essentially allows certain activation functions to express themselves more prominently, or less prominently based on their attribution to the overall computed cost of the layer. Ideally, the activation functions will either be fully blocked which corresponds to a 0 or fully expressed which corresponds to a 1. This would allow a more restricted, but more easily optimized subset of linear combinations of activation functions. In order to achieve this behavior, as well as to allow the gate parameters to be optimized, the gate limiting function, l, is defined as the sigmoid function in (2). This function was chosen due to its relatively simple to compute derivative as well as its nonlinear behavior which constrains the activation functions to be fully suppressed or fully excited. This also prevents the gate values from growing uncontrollably.

$$l(x) = \frac{1}{1+e^{-x}} \tag{2}$$

Given the definition of a multiple activation function gate layer, its corresponding gradient update equations were derived in (3) and (4). The parameters are as follows: δg_i represents the gradient update to the gate values, while δu represents the gradient update to the input to the adaptive function layer.

$$\delta g_i = \delta v \frac{\partial l}{\partial g_i}(g_i) f_i(u) \tag{3}$$

$$\delta u = \sum_{i=1}^{N} \frac{\partial f_i}{\partial u}(u) l(g_i) \delta v \tag{4}$$

An additional cost term can be added to constrain the percentage of activation functions that are used in the network. This could allow for more restricted behavior to be modeled that prevents the activation functions from becoming unstable or growing out of control.

The gradient update can also be scaled in order to prevent the adaptive functions from taking over the optimization problem. This issue essentially is due to the fact that there are many more variables added to the overall cost of the network which must be optimized. In CNN's where there may be more activation parameters than filter or fully connected weight parameters, the latter two parameters can easily fall into the local minima due to the emphasis of the adaptive function parameters. In order to avoid this, a scaling factor S_f can be introduced which scales the entire gradient update. Additionally, a momentum term is recommended to avoid local minima.

An alternative method to avoid this problem is to vary the scale term per epoch such that certain epochs only train the network's original parameters, while other epochs will include the adaptive activation function parameters. This can be done by setting the scale factor to 0 for a certain number of epochs and then to any positive value for a specific number of epochs. Alternating in this fashion allows the adaptive activation functions to try to settle into the learned parameters of the network more naturally, however this needs to be explored further.

DOI: 10.2352/ISSN.2470-1173.2016.19.COIMG-149

Results

Experiments are performed on the CIFAR100 and CalTech256 datasets. The CIFAR100 dataset has 100 classes with 500 images for training and 100 images for testing respectively per class and has an image size of 32x32x3 pixels. The CalTech256 dataset has 257 classes, with 80 to 827 images per class with image size of 300x200x3 pixels.

Deep convolutional neural networks are used to contrast using traditional activation functions (baseline) as compared to the adaptive activation functions introduced in this paper. In order to obtain a fair comparison between baseline results and adaptive function results, the same training parameters were used for both methods without scaling the learning rates while training. To determine the best way of training the adaptive function networks, two separate trials were conducted: 1) replacing baseline activation functions with adaptive activation function layers after the first few layers of a deep CNN, and 2) replacing baseline activation function layers with adaptive activation function layers at the last few layers of a deep CNN. The results from each experiment support recent findings in neuroscience as well as related deep learning research.

Experiments show that although the ReLU activation function works best when used in isolation, the traditional sigmoid and tanh occur more frequently when multiple activation functions are allowed to become active at each node. Networks which support multiple activation functions per node are shown to significantly outperform networks with one activation function per node and also train significantly faster.

CIFAR100

Baseline accuracies were obtained through the use of a VGG-like architecture [21] as shown in Figure 2 using batch normalization [22] to speed up the training process. The maximum testing accuracy obtained without random translations or flips was 57.5% after 300 training epochs as shown in Figure 3.

In order to determine the best place in the network to apply adaptive functions to, two separate experiments were run with adaptive activation functions using the sigmoid, tanh, and rectified linear units to adapt to the network. In the first network, the adaptive activation functions were applied to the first six layers where previously rectified linear units were used. In this case, the overall training time of the network suffered and the testing accuracy dramatically dropped down to 51.3% at the end of 300 epochs. This is most likely due to the fact that the network was not able to generalize to the testing set with so many parameters to optimize in the first layers and therefore ended up doing poorly.

Next, the adaptive functions were applied to the last seven layers where previously rectified linear units were used. This configuration did much better than the previous and in fact produced a gain in accuracy of 2% when compared to the baseline case. Furthermore, the amount of epochs to reach the same accuracy for this adaptive case versus the baseline was much less. These results can be seen in Figure 3 which compares the baseline training and testing accuracies to the adaptive case.

From this comparison between the two methods, the adaptive functions seem to positively enhance the discriminative accuracy of a convolutional network when applied to the final layers of the network. It can be postulated that the early layers of the network are crucial in determining the overall accuracy of the network and therefore if too many extra parameters are included, the overall number of epochs for the network to converge increases. The overall accuracy is also impacted by this since there are undoubtedly many local minima introduced by the adaptive activation functions which can halt the progress of the overall network if techniques to avoid them are not used properly.

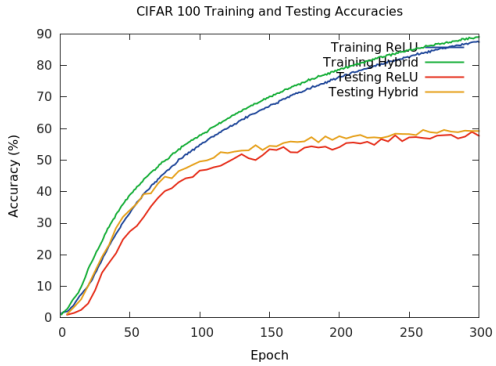

Figure 3: Accuracy comparison between baseline and adaptive functions for CIFAR 100 dataset.

The usage statistics of the adaptive networks are of importance to analyze since their layer-by-layer properties ultimately determine the success of the overall network. The total usage percentage of the final seven adaptive activation functions of the network are shown in Figure 4.

As shown by the layerwise usage statistics in Figure 4, the sigmoid activation function is used less than the other two functions in every case except for the third in which it ends up being used just as often as the hyperbolic tangent function. Surprisingly, the rectified linear unit function is only dominantly used in the final activation function layer between the two fully connected layers of the network. From these statistics, one can determine that the tanh function when paired with the rectified linear unit function will be used most often in most layers, however the sigmoid function can also become useful in order to clamp the output of the overall activation function.

A random sampling of activation functions from a single layer are shown in Figure 5 wherein each adaptive function is independently optimized for a single node.

Caltech256

Similar methods were used to obtain accuracies for the Caltech256 dataset. Images were resized to a size of 64x64 which necessitated a change in the number of layers in Figure 2. Two convolution layers and a max pooling were inserted before the fully connected layers in order to reduce the dimensions of the image down to a single pixel by the end of the network. The decision to reduce the size of each image so drastically from their original sizes ultimately cost the network significant accuracy, however for the purposes of comparison with the adaptive case, this is not that important. The baseline accuracy of the model utilizing only rectified linear unit activation functions was 31.1% as shown in Figure 6. The placement of these adaptive functions ultimately decided the overall general improvement or degradation of accuracies in the overall model.

Two separate model strategies were adapted from the the CIFAR 100 experimental findings. The first strategy substitutes

DOI: 10.2352/ISSN.2470-1173.2016.19.COIMG-149

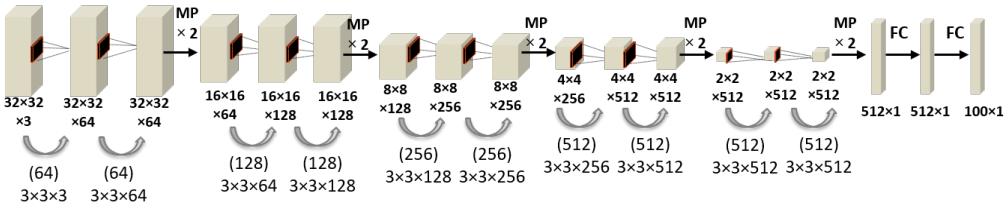

Figure 2: General convolutional network architecture for CIFAR 100 dataset.

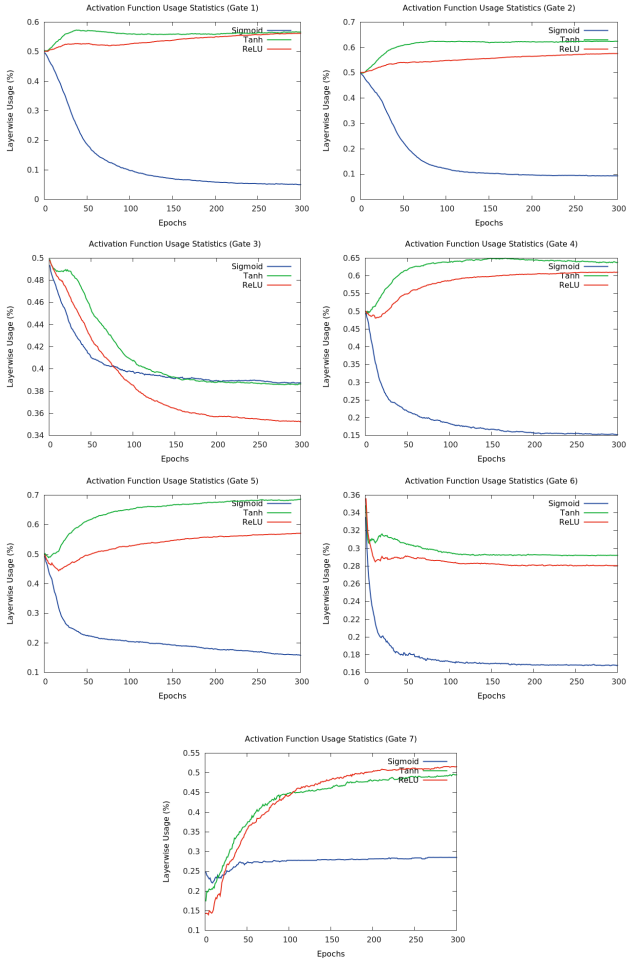

Figure 4: Activation function statistics per layer for the CIFAR 100 datset.

adaptive activation functions for the first four convolution layers, while the second strategy substitutes adaptive activation functions for the last five layers of the network. In the first case, both the training and testing accuracies were negatively impacted and after the same 300 training epochs, the final testing accuracy ended up at 28.3%. This mirrors the behavior of the CIFAR 100 experiments. This problem could most likely be avoided by scaling the gradient update to the adaptive functions, however this would result in comparatively longer training times.

The second experiment shows a minor boost in both training and testing accuracies over the same number of epochs, however the overall testing accuracy was around the same as the baseline by the end of 300 epochs as shown in Figure 6. Both of the adaptive case curves are bowed outwards and to the left of the baseline case clearly indicating an advantage in training time reduction. Although the overall accuracies are the same after 300 epochs, this most likely is a limitation of the amount of data that is being processed in the network itself rather than a limitation of the adaptive functions. If the images passed into the network were considerably larger and the network architecture itself was deeper, the accuracies may improve further for the adaptive case for the same amount of time. This is due to the fact that the the network would generalize more readily to the test set if less information was thrown away from the original images. Larger images would mean that the testing accuracies would take longer to saturate and the boost in accuracies seen in the current model would be able to increase over a longer duration of time. It takes roughly 150 epochs for the current model's testing accuracies to saturate in both the baseline and adaptive case.

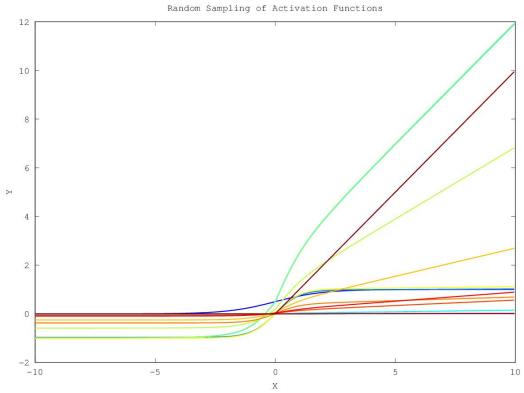

Figure 5: Random sampling of activation functions for CIFAR 100 dataset.

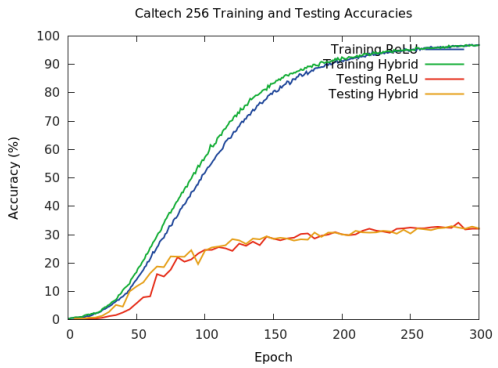

Figure 6: Accuracy comparison between baseline and adaptive functions for Caltech 256 dataset.

DOI: 10.2352/ISSN.2470-1173.2016.19.COIMG-149

Conclusion

This paper has shown that adaptive activation functions can shorten training time and increase classification accuracies when utilized in the later layers of a deep convolutional neural network. However further research must be done to determine strategies for reducing the negative impact that adaptive activation functions have on early layers of convolutional neural networks. There are also a large variety of activation functions that have yet to be tested using this adaptive method including non-monotonic functions which may be of interest to determine if they provide any advantage.

Acknowledgments

We greatly appreciate the NVIDIA corporation for their donation of the GPUs used in this research.

References

[1] Dayan, Peter, and Laurence F. Abbott. Theoretical neuroscience. Vol. 806. Cambridge, MA: MIT Press, 2001.

[2] Dawson, Michael RW, and DON P. SCHOPFLOCHER. "Modifying the generalized delta rule to train networks of non-monotonic processors for pattern classification." Connection Science 4.1 (1992): 19-31.

[3] Garabedian, Catherine E., et al. "Band-pass response properties of rat SI neurons." Journal of neurophysiology 90.3 (2003): 1379-1391.

[4] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks." International Conference on Artificial Intelligence and Statistics. 2011.

[5] Hara, Kazuyuki, and K. Nakayamma. "Comparison of activation functions in multilayer neural network for pattern classification." Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on. Vol. 5. IEEE, 1994.

[6] G. E. Hinton, S. Osindero, and T. Yee-Whye, "A fast learning algorithm for deep belief nets," Neural Computation, vol. 18, pp. 1527-54, 07/ 2006.

[7] Liao, Xiaofeng, et al. "Hopf bifurcation and chaos in a single delayed neuron equation with non-monotonic activation function." Chaos, Solitons & Fractals 12.8 (2001): 1535-1547.

[8] Merkel, Cory, Dhireesha Kudithipudi, and Nick Sereni. "Periodic activation functions in memristor-based analog neural networks." Neural Networks (IJCNN), The 2013 International Joint Conference on. IEEE, 2013.

[9] A.-R. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," IEEE Transactions on Audio, Speech and Language Processing, Vol. 20, pp. 14-22, 2012.

[10] Nakayama, Kenji, and Moritomo Ohsugi. "A simultaneous learning method for both activation functions and connection weights of multilayer neural networks." Proc. IJCNN. Vol. 98. 1998.

[11] Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010.

[12] Kang, Miao, and Dominic Palmer-Brown. "An adaptive function neural network (ADFUNN) for phrase recognition." Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on. Vol. 1. IEEE, 2005.

[13] Scheler, Gabriele. "Memorization in a neural network with adjustable transfer function and conditional gating." arXiv preprint q-bio/0403011 (2004).

[14] Scheler, Gabriele. "Regulation of neuromodulator receptor efficacy:implications for whole-neuron and synaptic plasticity." Progress in Neurobiology 72.6 (2004): 399-415.

[15] Soltiz, Michael, et al. "Memristor-based neural logic blocks for nonlinearly separable functions." Computers, IEEE Transactions on 62.8 (2013): 1597-1606.

[16] Vecci, Lorenzo, Francesco Piazza, and Aurelio Uncini. "Learning and approximation capabilities of adaptive spline activation function neural networks." Neural Networks 11.2 (1998): 259-270.

[17] Wong, K. W., C. S. Leung, and S-J. Chang. "Use of periodic and monotonic activation functions in multilayer feedforward neural networks trained by extended Kalman filter algorithm." Vision, Image and Signal Processing, IEEE Proceedings-. Vol. 149. No. 4. IET, 2002.

[18] Xu, Shuxiang, and Ming Zhang. "Justification of a neuron-adaptive activation function." ijcnn. IEEE, 2000.

[19] Hornik, Kurt and Stinchcombe, Maxwell and White, Halbert. "Multilayer feedforward networks are universal approximators." Neural Networks, Vol. 2, pp. 359-366, 1989.

[20] Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems, pp. 1097-1105, 2012.

[21] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." CoRR, abs/1409.1556, 2014.

[22] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." CoRR, abs/1502.03167, 2015.

[23] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." arXiv:1502.01852, 2015.

Author Biography

Michael Dushkoff is a BS/MS Computer Enginerring student from the Rochester Institute of Technology. He has worked as a researcher in the field of machine learning in the RIT Machine Intelligence Lab since 2014. His research is focused on brain-inspired computational systems and taking inspiration from biological systems to improve current generation technologies.