A New Activation Function for Training Deep Neural Networks to Avoid Local Minimum

Abhinav Sagar

Vellore Institute of Technology, Vellore, Tamil Nadu, India

ABSTRACT

Activation functions have a major role to play and hence are very important while training neural networks. Understanding various activation functions and their advantages and disadvantages is crucial to achieve better results. This paper will first introduce common types of non linear activation functions and then evaluate their characteristics with their pros and cons. We will be focussing only on deep neural networks as it has proven to be much more difficult to train while avoiding overfitting. We have proposed a new activation function named - Abhinav which adds a non linearity with parametric coefficients to the Swish activation function. This has proven to give better results on the MNIST dataset. We reason this is because the model avoids getting stuck in local minima due to only the sigmoidal term present in the Swish function. The coefficients are automatically adjusted in each and every iteration i.e. coefficient values are reduced if the error is large and also sometimes reducing it to zero thus removing coefficients present in the polynomial. This new activation function can be made to generalize to other tasks including multi class and multi label image classification also.

***Index Terms*—Neural Networks, Deep learning, Image classification, Activation function, Swish**

1. INTRODUCTION

Deep neural network algorithms are multi layered learning techniques that allows non-linear nodes to transform representations from the raw input into the higher levels of abstract representations, with many of these transformations producing learned complex functions. This is very important as most of the real world problems require a nonlinear treatment. The deep learning research was inspired by the limitations of the conventional learning algorithms especially being limited to processing data in raw form, and the human learning techniques by changing the weights of the simulated neural connections on the basis of experiences, obtained from past data.

The artificial neural networks (ANN) are biologically inspired computer networks, designed by the inspiration of the workings of the human brain. These ANNs are called networks because they are composed of different functions in each and every layer, which gathers knowledge by detecting the relationships and patterns in data using past experiences known as training examples. The learned patterns in data are modified by an appropriate activation function and presented as the output of the neuron as shown in Fig 1.

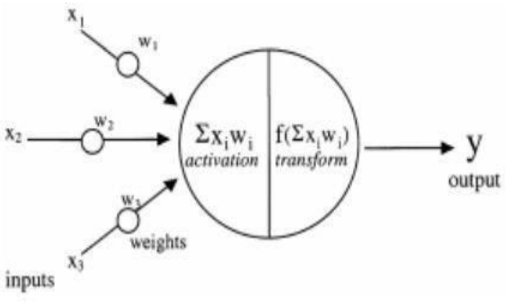


Fig 1. Architecture of neural network

So basically inputs coming from the previous layers are multiplied by the weights and a bias term is added to it. Then an activation function is used on top of it to convert the above function to a nonlinear function and represent the problem we are trying to solve. In every iteration, the loss is calculated which is the difference between the actual output and the predicted output. This loss is used to train the neural network by backpropagation algorithm by calculating the gradient of the loss function with respect to the weights and biases present in the network.

The most popular activation function for training neural networks that was widely used was the Sigmoid function however, when the Rectifier Linear Unit (ReLU) (Hinton et al) was introduced, it soon became a better and since then has been considered as the state of the art activation function. The reason is that it has been proven to be successful across various domains like image classification, image segmentation, language processing, recommender systems, etc. Later on, different variants of the ReLU activation function have been introduced and this experiment explores them and their impact on the MNIST (Modified National Institute of Standards and Technology) digits set. The MNIST (LeCun et al) dataset has 50,000 training images, 10,000 validation images, and 10,000 test images, each showing a 28×28 grey-scale pixel image of one of the 10 digits.

Activation functions are functions used in neural networks to computes the weighted sum of input and biases. This is used to decide if a neuron can be fired or not. It manipulates the presented data through gradient descent and afterwards produce an output for the neural network containing the parameters present in the data. The need for activation functions include to convert the linear input signals and models it as non-linear output signals, which aids the learning of high order polynomials beyond a single degree for training deep neural networks. A special property of the non-linear activation functions is that they are differentiable which is very important. This makes them work while training using backpropagation algorithm. The deep neural network is a neural network with multiple hidden layers and output layer. A typical block diagram of a deep learning model is shown in Figure 2.

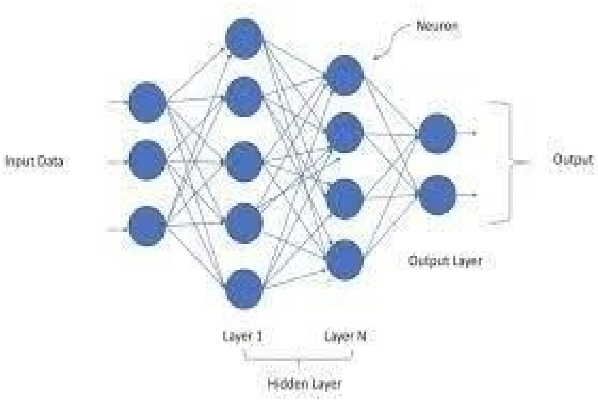


Fig 2. Architecture of deep neural network

The input layer accepts the data for training the neural network which comes in various formats from images, videos, texts, speech, sounds, or numeric data, while the hidden layers are made up of mostly the convolutional and pooling layers (Krizhevsky et al), of which the convolutional layers detect the local patterns and features present in the data from the previous layers, presented in array-like forms for images while the pooling layers semantically merges similar features into one. The output layer contains an activation function to add the nonlinearity to neural network, specially to perform classifications or predictions with associated probabilities. The position of the activation function in a network structure depends on its function in the network. The activation function is placed after the hidden layers, and hence it is used to convert the learned linear mappings into non-linear forms for propagation. The propagated weights are used in the output layer to perform predictions.

2. Deep Learning

Deep learning is a set of machine learning methods that was inspired by information processing and distributed communication in networks of biological neurons. Deep learning predominantly involves development, training and utilization of artificial neural networks (ANNs). ANNs are networks of artificial neurons that are based on biological neurons. Every ANN has at least 3 layers: an input layer that takes the input, a hidden layer that trains on the dataset fed to the input layer, and an output layer that gives an output depending on the application.

Deep learning has been gaining wide popularity because of the results it achieves that have never been seen in any other machine learning method. Convolutional Neural Networks (Krizhevsky et al), are extensively used for image-based applications, have achieved better results than humans in object detection and classification, semantic segmentation and instance segmentation.

3. Activation Functions

Activation functions (Agostinelli et al) are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be fired or not, based on whether each neuron's input is relevant for the model's prediction. The other purpose of the activation functions is to normalize the output of each neuron to a range between 0 and 1 or between -1 and 1. Also activation functions must be computationally efficient (Pedamonti et al) because they are calculated across thousands or even millions of neurons for each data sample. Modern neural networks use a technique called backpropagation to train the model, which places an increased computational strain on the activation function, and its derivative function. This is why activation function should be differentiable. Modern neural network use non-linear activation functions to model real world problems which cannot be solved using linear activation functions. They allow the model to create complex mappings between the network's inputs and outputs, which are essential for learning and modeling complex data, such as images, video, audio, and data sets which are non-linear or has high dimensionality. More than 80% of the real world datasets comes under this category and hence these activation functions are very important.

In this paper, we have compared Sigmoid, Hyperbolic tangent, ReLU, LeakyReLU, Swish and our own activation function. Other activation functions which have been successful are image based tasks are Elu (D. A. Clever et al) and S shaped ReLU (X. Jin et al). A comparative study of previous state of the art activation functions can be found in (Forest Agostinelli et al) and P. Ramachandran (et al).

4. DATASET

The MNIST (LeCun et al) database of handwritten digits, available has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image. It is a dataset of 60,000 small square 28×28 pixel grayscale images of handwritten single digits between 0 and 9. The task is to classify a given image of a handwritten digit into one of 10 classes representing integer values from 0 to 9, inclusively. The sample images from the dataset is shown in Fig 3.

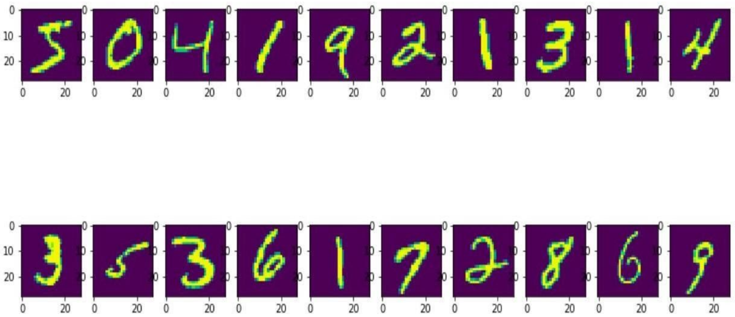
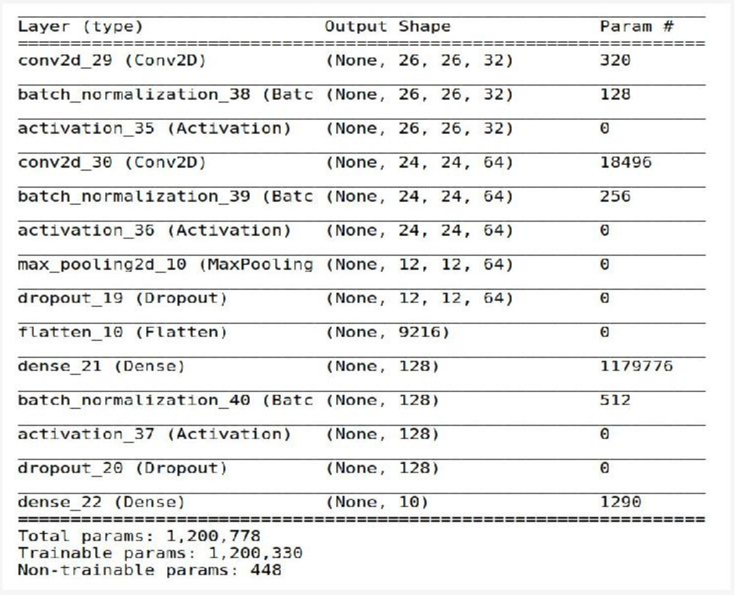


Fig 3. MNIST images from the dataset

5. PROPOSED METHOD

We have used various activation functions for training deep neural networks and made a comparative study of the results achieved. We used a batch size value of 128, learning rate value of 1e-5 and trained the neural network for 20 iterations. The mentioned hyper-parameters are constant throughout the study.

First we normalized the pixel values for every image in the dataset so that each pixel value has a value between 0 and 1. The neural network that we used is a feedforward convolutional neural network like shown in the ImageNet paper (Krizhevsky et al) having convolutional layers, various activation layers and max-pooling layers. In between we also used batch normalization (Sergey Ioffe et al) layers to improve the speed, performance, and stability while training deep neural networks. It’s main function is to normalize the input layer by adjusting and scaling the activations. Also we have used 25% dropouts (Hinton et al) to reduce overfitting. Finally we have used fully connected layer with softmax as the activation function as this is a classification problem. We used Adam as the optimizer (Kingma et al) and categorical cross entropy as the loss function for training the neural network. The complete architecture of the neural network is shown in Fig 4.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_29 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
batch_normalization_38 (Batc (None, 26, 26, 32)        128
_________________________________________________________________
activation_35 (Activation)   (None, 26, 26, 32)        0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 24, 24, 64)        18496
_________________________________________________________________
batch_normalization_39 (Batc (None, 24, 24, 64)        256
_________________________________________________________________
activation_36 (Activation)   (None, 24, 24, 64)        0
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 12, 12, 64)        0
_________________________________________________________________
dropout_19 (Dropout)         (None, 12, 12, 64)        0
_________________________________________________________________
flatten_10 (Flatten)         (None, 9216)              0
_________________________________________________________________
dense_21 (Dense)             (None, 128)               1179776
_________________________________________________________________
batch_normalization_40 (Batc (None, 128)               512
_________________________________________________________________
activation_37 (Activation)   (None, 128)               0
_________________________________________________________________
dropout_20 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_22 (Dense)             (None, 10)                1290
=================================================================
Total params: 1,200,778
Trainable params: 1,200,330
Non-trainable params: 448
```

Fig 4. Neural network architecture at a glance

The same experiments were repeated with the only change being the activation function. We have compared various state of the art activation functions like sigmoid, hyperbolic tangent, ReLU, Leaky ReLU and Swish. Also we tested with our own activation function which we named as Abhinav.

6. Activation

Functions Sigmoid

Function

Sigmoid suffers from the vanishing gradient problem ie for very high or very low values of input values, there is almost no change to the prediction, causing a vanishing gradient problem. This can result in the network refusing to learn further, or being too slow to reach an accurate prediction. Also the outputs are not zero centered and it is very computationally expensive to calculate the gradients for every iteration. The activation function plot of Sigmoid is shown in Fig 5.

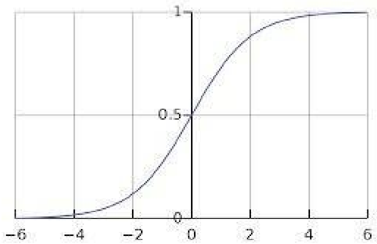


Fig 5. Sigmoid Function

Hyperbolic Tangent Function

This activation function is zero centered thus making it easier to model inputs that have strongly negative, neutral, and strongly positive values. It has the same problems as encountered by the sigmoid ie it suffers from vanishing gradient problem and is computationally expensive to calculate the gradients. The activation function plot of hyperbolic tangent is shown in Fig 6.

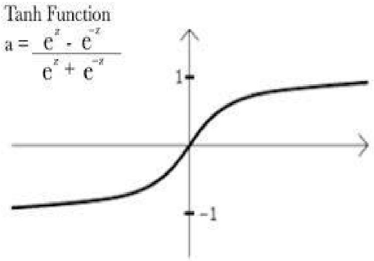


Fig 6. Hyperbolic Tangent Function

ReLU Function

ReLU (Agarap et al) is by far the most popular activation function for training deep neural networks. The reason for its popularity is that it is computationally efficient ie allows the network to converge very quickly. It suffers from dying relu problem when ReLU neurons become inactive and only output 0 for any input. A lot of work has been done to prevent this problem, the most famous one being changing the initialization strategy for the neural network by adding randomness. The activation function plot of ReLU function is shown in Fig 7.

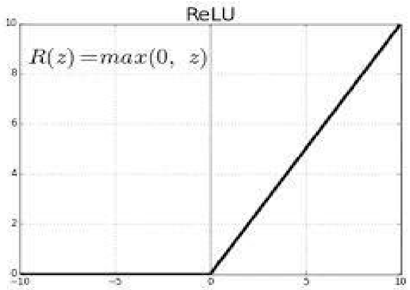


Fig 7. ReLU Function

Leaky-ReLU Function

This activation function prevents dying relu problem. This variation of ReLU has a small positive slope in the negative area, so it does enable backpropagation, even for negative input values. The problem with Leaky Relu is that it does not provide consistent predictions for negative input values. The activation function of Leaky ReLU plot is shown in Fig 8.

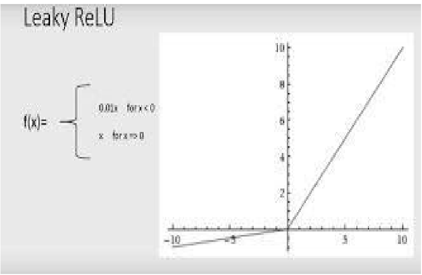


Fig 8. Leaky ReLU Function

Swish Function

Swish is a new, self-gated activation function. It performs better than ReLU with a similar level of computational efficiency for many problems. The problem with Swish is that the results are not generalizable ie although it works for simpler problems but fails for more complex problems involving multi class and multi label classification. The activation function plot of Swish function is shown in Fig 9.

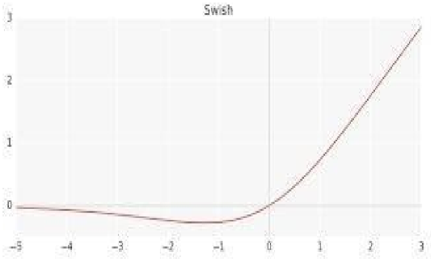


Fig 9. Swish Function

Abhinav Function

In this paper, we have proposed Abhinav activation function to combat the problems associated with the previous activation functions. Abhinav adds a non linearity term to the Swish activation function with parametric coefficients. While training the neural networks, the coefficients are either penalized or magnified depending on the error in each iteration. By this method, we get to train the neural network with best of both Swish as well as a parametric nonlinear function. The activation function plot of Abhinav function is shown in Fig 10.

The results achieved on the MNIST dataset is better than all the previous state of the art activation functions. We reason this is due to avoidance of local minima and the model is able to converge to the global minima due to addition of the parametric nonlinear terms whose coefficients are adjusted in every iteration to yield better results. This leads to better generalization also ie better results for multi class and multi label image classification as well as other problems.

The activation function plot of Abhinav function is shown in Fig 11. The two curved lines denote the ranges which can be covered by the function due to the addition of non linear term. For negative values of x, y is not bounded and the coefficient of the nonlinear terms present in polynomial is adjusted in every iteration while training the neural network.

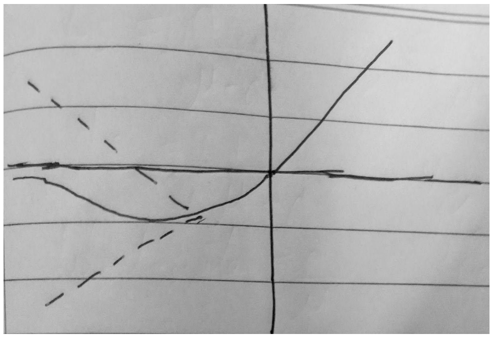
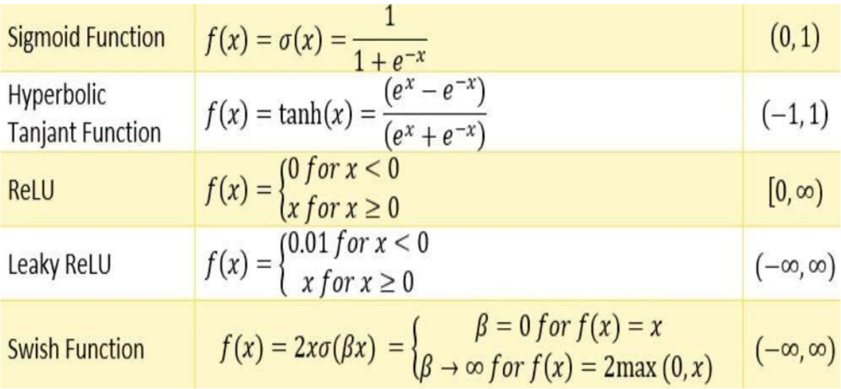


Fig 11. Abhinav activation function

As a summary, the equations and the range of output values of various activation functions are shown in Fig 12.



Sigmoid Function	$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$	$(0,1)$
Hyperbolic Tanjant Function	$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$	$(-1,1)$
ReLU	$f(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$	$[0,\infty)$
Leaky ReLU	$f(x) = \begin{cases} 0.01 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$	$(-\infty,\infty)$
Swish Function	$f(x) = 2x\sigma(\beta x) = \begin{cases} \beta = 0 \; for \; f(x) = x \\ \beta \rightarrow \infty \; for \; f(x) = 2\max(0,x) \end{cases}$	$(-\infty,\infty)$

Fig 12. Comparing various state of the art activation functions

7. EXPERIMENTAL RESULTS

The validation loss and validation accuracy as a function of epochs are shown in Fig 13.

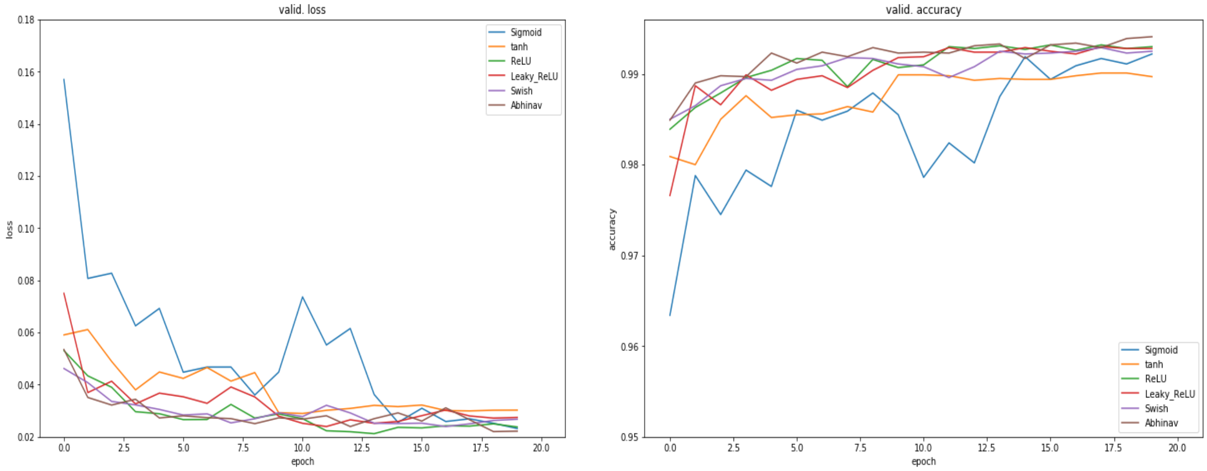


Fig 13. Validation loss and validation accuracy vs epochs

Next, we compare the final results ie the loss and the accuracy on the validation set. Clearly the new activation function Abhinav performed better than all the state of the art activation function up until now published in literature. The final results are shown in Fig 14.

Activation function	Validation accuracy	Validation loss
Sigmoid	0.9866	0.0567
Hyperbolic Tangent	0.9905	0.0474
ReLU	0.9929	0.0401
LeakyReLU	0.9925	0.0403
Swish	0.9918	0.0404
Abhinav	**0.9942**	**0.0387**

Fig 14. Final results ie validation accuracy and validation loss achieved using various activation functions

8. CONCLUSION

This paper provides a comprehensive summary of activation functions used while training deep neural networks and highlights the current trends in the use of these functions in practice. We first presented a brief introduction to deep learning and activation functions, and then outlined the different types of activation functions discussed, with some specific applications where these functions were used in the development of deep learning based architectures and systems. The activation functions have the capability to improve the learning of the patterns in data thereby automating the process of features detection and justifying their use in the hidden layers of neural networks, and usefulness for classification purposes across various domains of machine learning. We mentioned the advantages and disadvantages of various activation functions while training deep neural networks. Finally we proposed activation function named as Abhinav which adds a nonlinearity to the sigmoid term present in the Swish activation function. This led to better results than the state of the art on MNIST dataset. An important point to be noted is that there are other activation functions also that has not been discussed in this literature as we focused on the activation functions, used in deep learning applications.

REFERENCES

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[2] Mart´ın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In USENIX Symposium on Operating Systems Design and Implementation, volume 16, pp. 265–283, 2016.

[3] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. arXiv preprint arXiv:1412.6830, 2014.

[4] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In International Conference on Machine Learning, 2013.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision, pp. 1026–1034, 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016a.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning, pp. 448–456, 2015.

[8] . K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks. arXiv preprint arXiv:1603.05027v3,2016.

[9] Xu, L., Ren, J. S., Liu, C., & Jia, J. (2014). Deep convolutional neural network for image deconvolution. In Advances in neural information processing systems (pp. 1790-1798).

[10] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In 2009 IEEE 12th International Conference on Computer Vision, 2009.

[11] . K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385,2015.

[12] Kamavisdar, P., Saluja, S., & Agrawal, S. (2013). A survey on image classification approaches and techniques. International Journal of Advanced Research in Computer and Communication Engineering, 2(1), 1005-1009.

[13] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.

[14] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[15] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[16] Gunter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. arXiv preprint arXiv:1706.02515, 2017.

[17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Technical report, University of Toronto, 2009.

[18] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In International Conference on Machine Learning, volume 30, 2013.

[19] . G. Huang, Z. Liu, K. Q. Weinberger and L. Maaten. Densely Connected Convolutional Networks. arXiv:1608.06993v3,2016.

[20] . G. Huang, Y. Sun, Z. Liu, D. Sedra and K. Q. Weinberger. Deep Networks with Stochastic Depth. arXiv:1603.09382v3,2016.

[21] D. Pedamonti, “Comparison of non-linear activation functions for deep neural networks on MNIST classification task,,” arXiv, 2018.

[22] Suo Qiu and Bolun Cai. Flexible rectified linear units for improving convolutional neural networks. arXiv preprint arXiv:1706.08098, 2017.

[23] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Quoc Le, and Alex Kurakin. Large-scale evolution of image classifiers. arXiv preprint arXiv:1703.01041, 2017.

[24] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In International Conference on Machine Learning, pp. 2217–2225, 2016.

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[26] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:1505.00853, 2015.

[27] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. arXiv preprint arXiv:1707.07012, 2017.

[28] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for Activation Functions,” ArXiv, 2017.

[29] C. Bircanoglu and N. Arica, “A comparison of activation functions in artificial neural networks,,” in Signal Processing and Communications Applications Conference (SIU), vol. 26. IEEE‘, 2018, pp. 1–4.

[30] A. Maas, A. Hannun, and A. Ng, “Rectifier Nonlinearities Improve Neural Network Acoustic Models,” in International Conference on Machine Learning (icml), 2013.

[31] B. Xu, N. Wang, H. Kong, T. Chen, and M. Li, "Empirical Evaluation of Rectified Activations in Convolution Network," arXiv, 2015.

[32] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan, "Deep Learning with S-shaped Rectified Linear Activation Units,," arXiv, pp. 1737–1743, 2015.

[33] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs),"

[34] L. Trottier, P. Giguere, and B. Chaib-draa, "Parametric Exponential Linear Unit for Deep Convolutional Neural Networks,"

[35] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," arXiv, 2013.