Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

Training Neural Networks with Multi-Activations

Nitesh Nijhawan
Computer Science
SRM Institute of Science and Technology,
Chennai, India

Siddhartha Dhar Choudhury
Software Engineer
Avalara
Chennai, India

***Abstract*—An activation function helps a neural network understand and map non linear relationship between data points. These functions are applied to each node of a network to transform the values from a linear space to a non-linear one. These functions thus help in better learning features of a real world problem which can seldom be linear in nature. Choosing an activation function has been an area of continual research and certain functions are found to work well when applied to a problem. Some popular functions are - ReLU, Sigmoid, Softmax, and Tanh, these tend to work well under constrained settings and have helped achieve state-of-the-art results with neural networks on a wide array of applications. Commonly, a deep neural network contains several layers each communicating with the adjacent layers to get important features during forward propagation and in return helping them with better parameter values using chain rule in backpropagation. In a way these activation functions help the information pass from one layer to the next, when this is compared to a human network (with people interacting with one another) there are significant differences. With the activation functions the message passed to the succeeding layer is either bounded between certain limits or consists of a set of discrete values, human communication on the other hand can be considered as a mixture of both of these. Hence in this work, we intend to borrow ideas from human communication and apply multiple activation functions in a single layer of a deep neural network so that the features learnt by the nodes are passed more efficiently to the subsequent layer.**

I. INTRODUCTION

The field of Deep Learning is evolving and neural networks are now being used to solve most of the problems which were once solved using complex Machine Learning models. The major reason behind the change is the Neural Network's success, it's ability to learn the complex relations which maps the input to the output. In majority of cases the relation between input and output are too complex to be derived.

A neural Network is formed by attaching finite number of layers each of which contains a finite number of perceptrons. The Developer makes the entire structure of the network with a large number of unknown parameters and the process of training these parameters is done by understanding the relation between input and output values.

There is communication of data between consecutive layers of the network. The communication consists of getting the data, manipulating it, and giving it to the next layer as input. the manipulation consists of passing the data from a function.

These functions are applied to the input of each perceptron of the network and it transform the values from a linear space to a non-linear one. These functions thus help in better learning features of a real world problem which can seldom be linear in nature. These activation functions play the key role in finding the complex non-linear relations between input values and output labels.Some popular functions are - ReLU, Sigmoid, Softmax, and Tanh.

The trends in functioning of successive layer has been an area of continual research and on basis of the outcomes it can be concluded that as the layers progress, the network tends to extract more significant information out of the input information, hence every next layer extracts and manipulate the information which it has got from the previous layer.

With respect to the above context we can comply to the fact that by improving the communication of data between the layers, we can have a better learning of the parameters. This Ideology forms the main theme of this paper.

We can compare the communication between two layers of neural network with the communication between two humans, and by doing this we can identify the factors of an efficient communication and we can accordingly use these factors to improve the communication between the neural network layers.

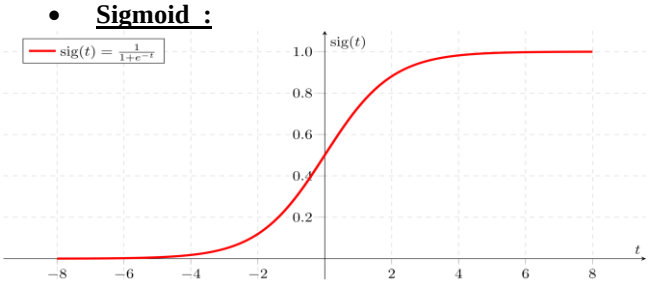
We make an effort to classify each activation function as a component of human conversation. A human conversation consists of a combination of several classification and regression and we tend to do the same with the activation functions in each layer of neural network.

II. UNDERSTANDING THE BASIC IDEOLOGY

The paper deals with the attempt to find a relation between the communication of data between two layers of neural network and the communication between two humans. We compare the components involved in both these communication and this leads us to find the common components.

Some of the components involved in human communication are True, False, and magnitude while the components involved in data communications are ReLU, Sigmoid, Softmax, and Tanh. Each of these activation function can be assumed to perform a particular task according to human conversation as :

- **Sigmoid :**

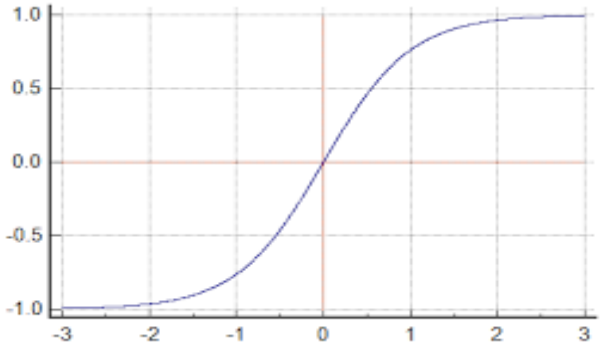


As we see from the representation of Sigmoid that it gives a value between 0 and 1 . A value close to 0 can denote False

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

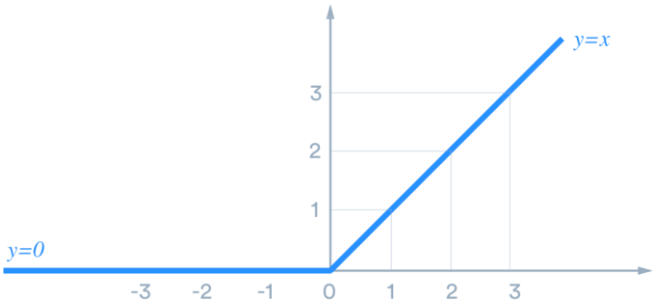
while a value near to 1 can denote True. Hence Sigmoid adds only True and False to the communication.

- **Tanh :**



As we see from the representation of Tanh that it gives a value between -1 and 1. A value close to -1 can denote False while a value near +1 can denote True and additionally a value near 0 can denote Neutral. Hence Tanh adds True, False and Neutral to the communication.

- **ReLU :**



As we see from the representation of ReLU that it gives a value between 0 and infinity. A value close to 0 can denote False while any other positive value denotes the magnitude which we want to convey. Hence ReLU adds False and magnitude to our conversation.

According to the general convention we use only one activation function at a time and this leads to the lack of components. In this paper we have come forward with the idea of putting together more than one type of activation function between two layers. By putting more than one type of activation function we can include more number of components into the communication of data from one layer to other.\\

It can mathematically be noted that because of involvement of different functions the non linearity increases . This increase in non linearity helps the function to learn more complex relations.

III. EXPERIMENTATION

The various experiments compare the three techniques of mixing activations functions mentioned in the previous section. We have tested our hypotheses on three standard datasets - CIFAR 10, Fashion MNIST and MNIST.

A Brief Descriptions of Datasets

Before diving into the details of the experiments it is necessary to understand the datasets and the problems they solve.

- **CIFAR10 :**

The CIFAR 10 contains images belonging to 10 classes - airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. The database contains 60,000 images with 6,000 images belonging to each of the aforementioned classes. The entire set is split into two parts - 50,000 images put in training set and the rest 10,000 in testing set. Each image is of dimension 32x32 and is a coloured (or RGB) image.

- **Fashion MNIST :**

Fashion MNIST contains 70,000 images of 10 different classes of cloth. Each image is a black and white image of dimension 28x28. The entire set is split into training set containing 60,000 images and testing set containing 10,000 images. The various classes in the dataset are - T-Shirt/Top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.

- **MNIST :**

MNIST dataset consists of 70,000 images belonging to 10 classes. This contains handwritten digits from 0 till 9 as 28x28 grayscale images. The entire set is split like Fashion MNIST with 60,000 images in the training set and 10,000 images in the testing set.

With a clear understanding of the data the model has been trained on, we are in a position to dive deeper into the experiments.

CIFAR 10

The basic structure of the convolution neural network used for training the model has two convolution layers each followed by a max pooling layer with a pool size of 2, this is followed by three dense layers of 120, 84 and 10 nodes each respectively. The final layer (generally referred to as the output layer) has ten nodes as there are 10 different classes of images in the CIFAR-10 database. The final layer is activated using Softmax function which converts the floating point Z values into a probability distribution.

The convolution layers use a five cross five (5x5) kernel size (size of filters). The first convolution layers increases the depth of the image from three to six, and the second one takes the six layer deep image and converts it into a eight layer deep image.

The base network uses ReLU (Rectified Linear Unit) activation function after each convolution and fully connected layers except the last one. The final layer is activated using softmax function. The choice of the final activation remains same for all the different variants of our proposed multi-activation networks, but the functions before the final output layer are modified. For the experiments we have made use of three different activation functions for the preceding layers -

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

sigmoid, tanh, ReLU. All the models were trained for five (5) training iterations (epochs).

Table 1:Training accuracy for various models for CIFAR 10

Model	Training Accuracy
Vanilla CNN	61%
Deterministic half (sigmoid, tanh)	57%
Deterministic triple (sigmoid, tanh, ReLU)	58%
Double layerwise alternate (sigmoid, tanh)	60%
Triple layerwise alternate (sigmoid, tanh, ReLU)	58%
Once stochastic double (sigmoid, tanh)	58%
Once stochastic triple (sigmoid, tanh, ReLU)	60%
Everytime stochastic double (sigmoid, tanh)	61%
Everytime stochastic triple (sigmoid, tanh, ReLU)	61%
Once stochastic layerwise double (sigmoid, tanh)	62%
Once stochastic layerwise triple (sigmoid, tanh, ReLU)	64%
Everytime stochastic layerwise double (sigmoid, tanh)	62%
Everytime stochastic layerwise triple (sigmoid, tanh, ReLU)	62%

MNIST

This model contains two fully connected networks. The first layers takes in input of 784 pixel values (from the 28x28 grayscale images flattened into vectors). The first layers has one thousand (1,000) nodes, this is followed by a softmax output layer consisting of 10 (ten) different nodes which is required for the 10 different classes of digits in the MNIST database. The final layer converts all the Z values to a probability distribution to denot the class a particular image belongs to, this is done using the softmax activation function.
The base network (which is used for comparison) uses ReLU (Rectified Linear Unit) activation function in first fully connected layers except the last one. The final layer is activated using softmax function. The choice of the final activation remains same for all the different variants of our proposed multi-activation networks, but the functions before the final output layer are modified. For the experiments we have made use of three different activation functions for the preceeding layers - sigmoid, tanh, ReLU. All the models were trained for five (5) training iterations (epochs). Since only a single layer has the flexibility of handling multiple activations, the layerwise techniques were not used in this.

Table 2:Training accuracy for various models for MNIST

Model	Training Accuracy
Vanilla NN	82%
Deterministic half (sigmoid, tanh)	77%
Deterministic triple (sigmoid, tanh, ReLU)	78%
Double layerwise alternate (sigmoid, tanh)	81%
Triple layerwise alternate (sigmoid, tanh, ReLU)	78%
Once stochastic double (sigmoid, tanh)	78%
Once stochastic triple (sigmoid, tanh, ReLU)	81%
Everytime stochastic double (sigmoid, tanh)	81%
Everytime stochastic triple (sigmoid, tanh, ReLU)	82%
Once stochastic layerwise double (sigmoid, tanh)	82%
Once stochastic layerwise triple (sigmoid, tanh, ReLU)	84%
Everytime stochastic layerwise double (sigmoid, tanh)	83%
Everytime stochastic layerwise triple (sigmoid, tanh, ReLU)	83%

Fashion MNIST

The model details are same as MNIST, Table 3 consolidates the training accuracy results.

Table 3:Training accuracy for various models for Fashion MNIST

Model	Training Accuracy
Vanilla NN	74%
Deterministic half (sigmoid, tanh)	70%
Deterministic triple (sigmoid, tanh, ReLU)	70%
Double layerwise alternate (sigmoid, tanh)	73%
Triple layerwise alternate (sigmoid, tanh, ReLU)	70%
Once stochastic double (sigmoid, tanh)	70%
Once stochastic triple (sigmoid, tanh, ReLU)	74%
Everytime stochastic double (sigmoid, tanh)	73%
Everytime stochastic triple (sigmoid, tanh, ReLU)	74%
Once stochastic layerwise double (sigmoid, tanh)	74%
Once stochastic layerwise triple (sigmoid, tanh, ReLU)	75%
Everytime stochastic layerwise double (sigmoid, tanh)	75%
Everytime stochastic layerwise triple (sigmoid, tanh, ReLU)	74%

IV. UNDERSTANDING THE EXPERIMENTATION

In the above sections we have given a brief about the 3 datasets which we have used for the experimentation purpose. The table attached with every dataset represents the accuracy which we get for the different models which we have used with the given dataset.For every dataset we have trained 13 models, in which 1 is a standard model while the 12 others being the experimental models made by us based on the ideology mentioned in the above sections. In our approach to a achieve an increase in accuracy over the existing models we formed 12 entirely different models and each model differ with each other and with the standard model on basis of the activation function being used in it.
A brief description to each of the 13 models (including the standard model) :

- **Vanilla NN :**

Vanilla NNs are composed of an input layer, an output layer, and an arbitrary number of “hidden” layers in between that are “fully connected” (i.e. each neuron in one layer is connected to each neuron in the next layer).
The output of each layer is fed through a nonlinear activation function. This trick is what gives a vanilla neural net its nonlinear descriptive powers and makes it fundamentally different from linear regression. Researchers initially favored sigmoid functions and in this case we have used sigmoid function for every perceptron in every layer.

- **Deterministic half (sigmoid, tanh) :**

Now here we make changes in the vanilla network and in this case we change the activation function in every layer.In each layer we split the number of nodes into 2 equal halves. To all the perceptrons in one half we give sigmoid activation function while to other half we give tanh activation function.

- **Deterministic triple (sigmoid, tanh, ReLU)**

Now here we make changes in the vanilla network and in this case we change the activation function in every layer.In each layer we split the number of nodes into 3 equal parts. To all the perceptrons in one part we give sigmoid activation function while to other part we give tanh activation function and to the third part we give ReLU activation function.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

- **Double layerwise alternate (sigmoid, tanh)**

Now here we make changes in the vanilla network and in this case we change the activation function in every layer.We change the activation function of the layers in an alternate fashion as to one layer we will give the sigmoid activation function while to its neighbour layer we will give the tanh activation function.This way every pair of consecutive layers will have different activation functions.

- **Triple layerwise alternate (sigmoid, tanh, ReLU):**

Now here we make changes in the vanilla network and in this case we change the activation function in every layer.We change the activation function of the layers in an alternate fashion as to one layer we will give the sigmoid activation function while to its neighbour layer we will give the tanh activation function and further to its neighbouring layer we will gibe ReLU activation function and this ways the cycle will repeat. Hence every triplet of consecutive layers will have different activation functions.

- **Once stochastic double (sigmoid, tanh)**

Now here we make changes in the vanilla network and in this case we change the activation function in every layer.During our first iteration of forward propagation through the model then on reaching each layer we will make a random split in the number of perceptrons present in that layer and now we will assign the perceptrons sigmoid or tanh based on the split proportion.

and we do all this for each layer only through the first feed forward, hence the model gets permanent after one complete feed forward**.**

- **Once stochastic triple (sigmoid, tanh, ReLU)**

Now here we make changes in the vanilla network and in this case we change the activation function in every layer.During our first iteration of forward propagation through the model then on reaching each layer we will make a random split to 3 parts in the number of perceptrons present in that layer and now we will assign the perceptrons sigmoid or tanh or ReLU based on the split proportions.

and we do all this for each layer only through the first feed forward, hence the model gets permanent after one complete feed forward.

- **Everytime stochastic double (sigmoid, tanh) :**

This technique has been inspired from dropout regularization techinique it now here the models activation functions are not fixed rather they keep changing with each feed forward of the network.

We do 'Once stochastic double (sigmoid, tanh)' for each feed forward of the neural network.

Therefor every time we will get a different split up for the same layer while the setup for each layer will change with each feed forward.

- **Everytime stochastic triple (sigmoid, tanh, ReLU) :**

This technique has been inspired from dropout regularization techinique it now here the models activation functions are not fixed rather they keep changing with each feed forward of the network.

We do 'Once stochastic triple (sigmoid, tanh, ReLU)' for each feed forward of the neural network.

Therefor every time we will get a different split up for the same layer while the setup for each layer will change with each feed forward.

- **Once stochastic layerwise double (sigmoid, tanh) :**

In this approach we split up on the number of layers in the network ie we randomly choose layers where we will use only sigmoid function and in the remaining layers we will use only tanh function. This choice of layers randomly is done only one time which is in the first forward pass.

When we are doing first feed forward then for each layer we will randomly choose that the layer should get sigmoid or tanh and then based on the choice the activation function is assigned and is fixed for the network.

- **Once stochastic layerwise triple (sigmoid, tanh, ReLU) :**

In this approach we split up on the number of layers in the network ie we randomly choose layers where we will use only sigmoid function and in the remaining layers we will further split up to randomly choose whether to use tanh or ReLU function. This choice of layers randomly is done only one time which is in the first forward pass.

When we are doing first feed forward then for each layer we will randomly choose that the layer should get sigmoid or tanh or ReLU and then based on the choice the activation function is assigned and is fixed for the network.

- **Everytime stochastic layerwise double (sigmoid, tanh) :**

In this approach we do 'Once stochastic layerwise double (sigmoid, tanh)' for every single feed forward motion ie every time we do feed forward, we choose the activation function for each layer randomly.

- **Everytime stochastic layerwise triple (sigmoid, tanh, ReLU) :**

In this approach we do 'Once stochastic layerwise triple (sigmoid, tanh, ReLU)' for every single feed forward motion ie every time we do feed forward, we choose the activation function for each layer randomly**.**

V. MATH

Use either the Microsoft Equation Editor or the MathType add-on (http://www.mathtype.com) for equations in your paper (Insert | Object | Create New | Microsoft Equation or MathType Equation). “Float over text” should not be selected.

$$\Delta R = R_p - R_{ap} = -\frac{1}{2}\frac{(R_\uparrow - R_\downarrow)^2}{R_\uparrow + R_\downarrow} \qquad (1)$$

Each formula should occupy one line. Consecutive numbers should be marked in brackets. All equations should be numbered (the numbers should be aligned at the right), and cited (1) in the text.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 03, March-2021

VI. CONCLUSION

Initially we formed an ideology to use different activation function in random ways to give us an improved accuracy and then went through multiple experimentations to verify the ideology. On basis of the results of the experiments we can conclude that the increase in accuracy for each of the 3 datasets ,used by us, can be given as :

CIFAR 10 :

The observed maximum percentage increase in accuracy is 3 percent. As we see that the accuracy given by vanilla NN is 61 percent while the accuracy offered by our mode 'Once stochastic layerwise triple (sigmoid, tanh, ReLU)' is 64 percent , hence an increase of 3 percent is concluded.

MNIST :

The observed maximum percentage increase in accuracy is 2 percent. As we see that the accuracy given by vanilla NN is 82 percent while the accuracy offered by our mode 'Once stochastic layerwise triple (sigmoid, tanh, ReLU)' is 84 percent , hence an increase of 2 percent is concluded.

Fashion MNIST :

The observed maximum percentage increase in accuracy is 1 percent. As we see that the accuracy given by vanilla NN is 74 percent while the accuracy offered by our mode 'Once stochastic layerwise triple (sigmoid, tanh, ReLU)' is 75 percent , hence an increase of 1 percent is concluded.

Hence we can conclude that our assumption of using different activation function has proven to provide an increase in accuracy with respect to the standard neural networks for the standard datasets. Meanwhile we also observe that among the 12 models which we experimented on, 'Once stochastic layerwise triple (sigmoid, tanh, ReLU)' proves to be the most efficient network, giving a rise in accuracy, hence we can use this technique in other standard neural networks to get an increase in accuracy.

ACKNOWLEDGMENT

We would like to thank Andrew Ng, Geoffrey Hinton, Sebastian Thrun for their deep learning courses in Coursera and Udacity from where we started our deep learning journey. We would like to acknowledge Shashank Pandey for stimulating our creativity and proof reading this paper to provide his valuable inputs.

REFERENCES

[1] B DasGupta, G. Schnitger, “The Power of Approximating: A Comparison of Activation Functions”. In Giles, C. L., Hanson, S. J., and Cowan, J. D., editors, Advances in Neural Information Processing Systems, 5, pp. 615-622, San Mateo, CA. Morgan Kaufmann Publishers, 1993M. Young, The Technical Writers Handbook. Mill Valley, CA: University Science, pp. 1023-1045, 1989.

[2] M.I. Jordan, “Why the logistic function? A tutorial discussion on probabilities and neural networks”. Computational Cognitive Science Technical Report 9503, Massachusetts Institute of Technology, 1995E. H. Miller, “A note on reflector arrays (Periodical style-Accepted for publication, but the doi number has been assigned),” IEEE Trans. Antennas Propagat., doi:10.4316/ieee.1959.3422, to be published.

[3] Y. Liu, X. Yao, “Evolutionary Design of Artificial Neural Networks with Different Nodes”. In Proceedings of the Third IEEE International Conference on Evolutionary Computation, pp. 570-675, 1996

[4] J. M. Sopena, E. Romero, R. Alqu´ezar, “Neural networks with periodic and monotonic activation functions: a comparative study in classification problems”. In Proceedings of the 9th International Conference on Artificial Neural Networks, pp. 323-328, 1999

[5] G. Cybenko, “Approximation by superposition of a sigmoidal function”. Mathematics of Control, Signals, and Systems, 2(4):303–314, 1987

[6] B. Ciocoiu, “Hybrid Feedforward Neural Networks for Solving Classification Problems”. Neural Processing Letters, 16(1):81-91, 2002 Agostinelli, Forest, Hoffman, Matthew, Sadowski, Peter, and Baldi, Pierre. Learning activation functions to im-prove deep neural networks. In International Conference on Learning Representations, 2015

[7] Chen, Justin. Combinatorially generated piecewise activation functions. arXiv preprint arXiv:1605.05216, 2016.

[8] Li, Hongyang, Ouyang, Wanli, and Wang, Xiaogang. Multi-bias non-linear activation in deep neural networks. In International Conference on Machine Learning, 2016.

[9] Scardapane, Simone, Scarpiniti, Michele, Comminiello, Danilo, and Uncini, Aurelio. Learning activation functions from data using cubic spline interpolation. arXiv preprint arXiv:1605.05509, 2016

[10] Srivastava, Nitish, Hinton, Geoffrey E, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfit-ting. Journal of Machine Learning Research, 15(1): 1929–1958, 2014.

[11] Zhou, Zhi-Hua, Wu, Jianxin, and Tang, Wei. Ensembling neural networks: many could be better than all. Artificial Intelligence, 137(1-2):239–263, 2002.