Neural Network from Scratch

Mohammad Mohaiminul Islam, Aregawi Halefom Berhe

I. INTRODUCTION

This report is generated as general work procedure for implementing a Neural Network from scratch. The objective of the work is to progress step by step from a single neuron binary classification model to multi neuron multi class classification model. Later a comparative analysis was carried out between the response of these model in terms of accuracy, training loss, validation loss and other characteristic properties. We have used the MNIST [1] dataset for handwritten digit for the evaluation of the models. Also the effects of hyper-parameters have been explored.

II. BACKGROUND

Artificial Neural Network (see fig. 1) are computing system inspired by biological neural network that makes up our brain. It is collection or network of units or node called Neuron which tries to mimic the human neurons.A neuron can receive a signal then processes it, generate output and can signal neurons connected next to it. This generated signal usually feed into a activation function . Let us discuss about each component of NN in this section.

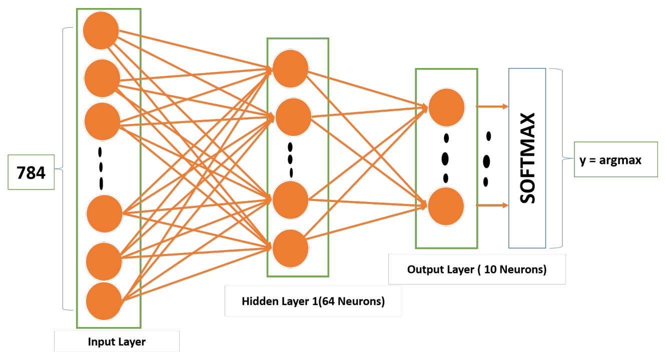

Fig. 1. Structure of an Artificial Neural Network.

A. Artifical Neurons

As shown in the Fig.2 artificial neuron is a mathematical function perceived as a model of biological neurons [2]. It receives input from some other units, or perhaps from an external source. Each input has an associated weight *w*, which can be modified so as to model synaptic learning. The unit computes the weighted sum of its inputs. For *N* number of input the unit computes following in (1) . Generally this sum from the neuron is then feed into an activation function. We discuss about activation function in next subsection.

$$y = \sum_{i=1}^{N} x_i w_i \tag{1}$$

Here w_i denotes weight of each input x_i.

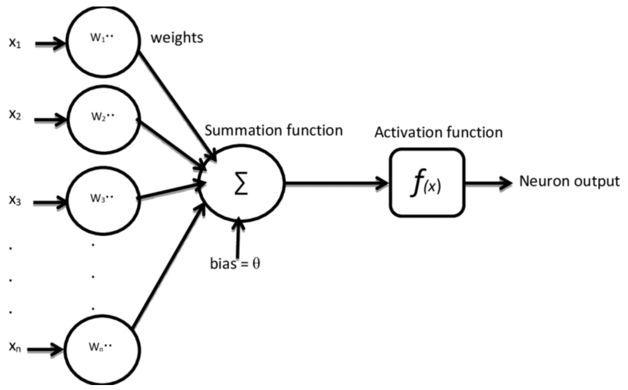

Fig. 2. Structure of an artificial neuron

Source : https://www.tutorialspoint.com/ann/artificial-neural-network-basic-concepts.htm

B. Activation Function

Activation functions are mathematical function that determine the output of a neuron.It 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 network's prediction. Activation functions also help normalize the output of each neuron to a certain range, generally between 1 and -1. Also have a major effect on the neural network's ability to converge and the convergence speed [3].

There are three main type of activation function that are used in the domain of ANN, e.g. Basic step function, Linear activation function and Non-linear activation function. However there exist some inherent problem with basic step and linear activation so in most cases we use non liner activation functions.Some of the well known non liner activation functions are Sigmoid ,Hyperbolic Tangent,Softmax ,ReLU (Rectified Linear Unit),Leaky ReLU etc. Here we will only discuss about sigmoid function as we are using this one in our project.

A sigmoid is a S-shaped mathematical function. Equation (2) show the formula for this function.The function is differentiable,that means, we can find the slope of the sigmoid curve at any two points.

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

C. Loss Functions

Loss function is a method for describing model performance.If the predicted label are totally off, the loss is going to be higher. If they're pretty good, the loss will be lower. Here the goal is to find the best set of parameters that minimizes the loss function as much as possible. So Loss helps us to understand how much the predicted value differ from actual value.

Loss function are often chosen depending on the given problem. As we have binary and multi class classification problem we will try to focus the loss function that are typically used for the following problems. Here we will discuss about cross entropy loss.

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

we can define the loss as (3). here n is the total number of items of training data, the sum is over all training inputs, x, p is the prediction and y is the corresponding desired output.

$$Loss = -\frac{1}{n}\sum_{x}[(y * log(p) + (1-y) * log(1-p))] \tag{3}$$

III. Training Optimization of the Network

In this section the process of training a neural network with forward pass and backward pass as well as how we optimize the our weight of the neurons have been briefly discussed

A. Training

- **Forward propagation.** Forward propagation (or forward pass) refers to the process of calculation and storing the intermediate variables for the network in order from the input layer to the output layer. It computes certain output for the every input data.
- **Backward propagation.** Backpropagation, short for "backward propagation of errors," is an learning approach for neural networks using gradient descent. Given an artificial neural network and a loss function, this method calculates the gradient of the error function with respect to the neural network's weights. Since neural networks contain a series layers of neurons, the activations propagates through different pathway.So chain rule is needed to calculate the gradient.Then the weights of the network is updated with the calculated gradients.The backward name comes from the fact the gradients are calculated backwards through the network.

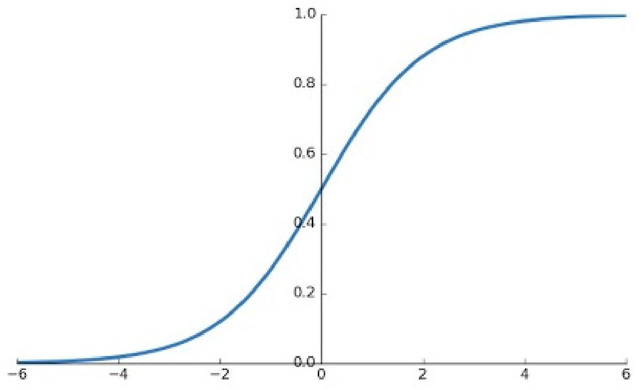

Fig. 3. Curve for sigmoid activation function

Source : https://www.geeksforgeeks.org/implement-sigmoid-function-using-numpy/

B. Optimization

Gradient descent is an optimization technique used for computing the model parameters (coefficients and bias) for neural networks. In this technique, we repeatedly iterate through the training set and update the model parameters in accordance with the gradient of the error with respect to the training set. Depending on the number of training examples considered in updating the model parameters, we have 3-types of gradient descents:

- **Batch Gradient Descent.** Parameters are updated after computing the gradient of error with respect to the entire training set. Although this takes a lot of time, it creates a smooth update of the weights.
- **Stochastic Gradient Descent.** Parameters are updated after computing the gradient of the error with respect to a single training example. This doesn't exploit the speed of using vectorized matrix multiplication as a single instance is considered in every update of the parameters. As a result, it creates a noisy update of the parameters.
- **Mini-Batch Gradient Descent** Parameters are updated after computing the gradient of error with respect to a subset (batch) of the training set. Depending on the batch size, the mini-batch gradient descent makes a compromise between the fast convergence and the noise associated with gradient updates, which makes it a more flexible and robust algorithm.

In our project we choose to use the mini-batch gradient descent, with a batch size of 128 for better result and faster convergence.

C. Weight Initialization

As the neural network is about finding the best parameters (weights and biases), which determine the network's performance. Careful weight initialization is crucial. It prevents layer activation outputs from exploding or vanishing during a forward pass through a deep neural network. If the weights are too large, the activation outputs will explode along the way. Likewise, if the weights are too small, the activation results will be insignificant to derivatives. Although there are several initialization techniques, we choose Xavier [5] initialization, which initializes the weights in a network by drawing them from a distribution with centered at zero mean and a specific variance given.

IV. Dataset

In the experiment we have used the dataset called MNIST[1].The MNIST dataset an acronym that stands for the Modified National Institute of Standards and Technology dataset that contains handwritten digits.Each image is of handwritten single digits between 0 and 9 as shown in the fig. 4.The digits have been size-normalized and centered in a fixed-size image. The resolution of the images are 2828 pixel and all the images are grayscale.It has a training set of 60,000 examples, and a test set of 10,000 examples.

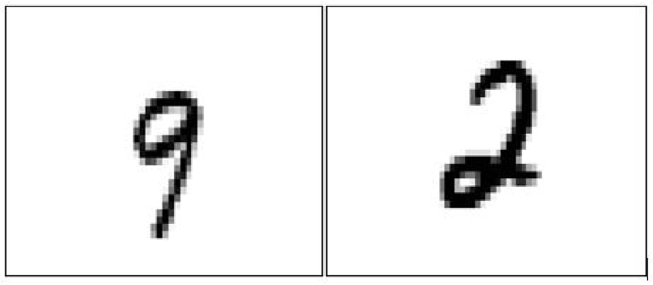

Fig. 4. Two sample images from the MNIST Dataset.

Further we split the training set into two set for training and validation of the model. Newly created training set contains 50000 images and validation set contains 10000 images.

V. Implementation

This project has been mainly written on python with the *numpy* package. Numpy was used for implementing the network core and calculation. For visualizing the results matlotib has been used Following are the system specification for the development of this project

- OS: Windows 10
- IDE: Visual Studio Code
- Language: Python : Numpy and Matplotlib

Three separate python file have been created for each of the problem in the project.

VI. Result & Analysis

In this section we will discuss about results and performance of three different variant of Neural network that we have built.

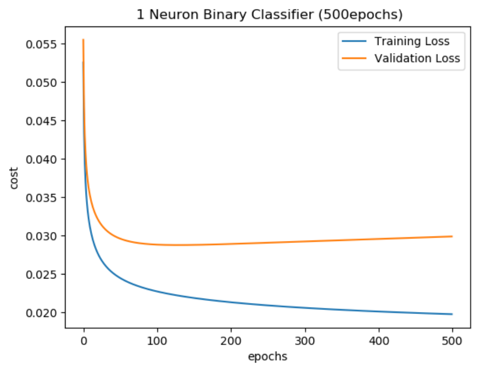

Fig. 5. Training and Validation losses for Single neuron model (Digit 0 vs. ALL)

A. Single Neuron Model

In this version of our NN we have only one neuron to solve our binary classification problem. Following are the hyper-parameter setting for the network ***Learning rate : 0.1, Momentum : 0.9, Batch Size : 128, Epoch : 500***

Al though we brought some changes to our MNIST dataset labels for the sake of our experiment. All the labels of the digit 0 (zero) have been encoded by label 1 and rest of the digits are encoded to label 0.Now we train our model for 500 epoch and accumulate the training and validation losses.From fig. 5 we see the training and validation loss curve. It can be noticed in the figure that, after just 20-30 epoch even though the training loss keep decreasing until 500 epoch , validation loss remained almost constant. Also after 200 epoch it started to increase instead of decreasing. This is a typical sign of model over fitting. Overfitting happens for few reasons, if we have a very complex model compared to the complexity or non-linearity of our data , if we over train our model etc. In our case since we have just a single neuron it is highly unlikely that, the overfitting is caused by model complexity. Here the most probable reason is over training a simple model. The model accuracy on the test data is 99.24 .

Next we wanted to explore the fact that, is some digit are harder to recognize than others for the model.So we put the digit 7 instead of zero this time. we encode the digit 7 (seven) with label 1 and rest of the digits are encoded to label 0. Again we accumulate the losses over each epoch and plot the losses in Fig. 6. Here the model is not quite overfitting for same model architecture and hyper-parameter setting. which means the complexity of data has to be higher so that for the same number of epoch we get less difference between training and validation losses. Also if we compare fig. 5 and fig. 6 the over all loss is higher for model trying to classify the digit 7. Hence it appears that the digit 7 is harder to classify than digit 0. The accuracy on the test data is 98.40.

Also if we look at the accuracy of the both experiment we

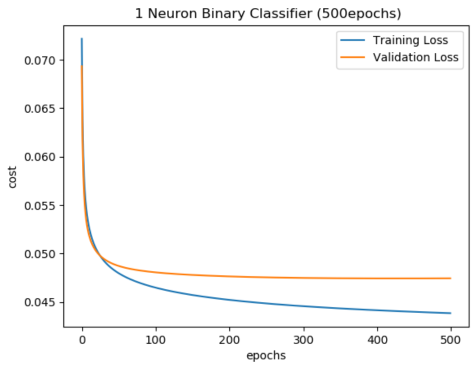

Fig. 6. Training and Validation losses for Single neuron model (Digit 7 vs ALL)

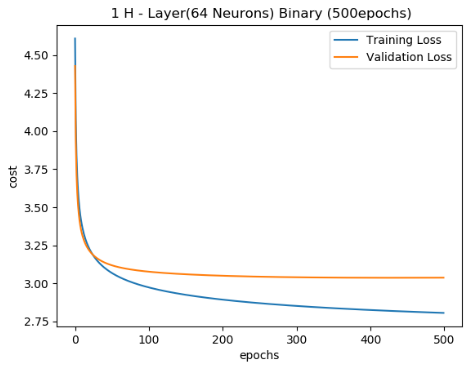

Fig. 8. Training and Validation losses for model with one hidden layer consisting of 64 neurons (Digit 7 vs ALL)

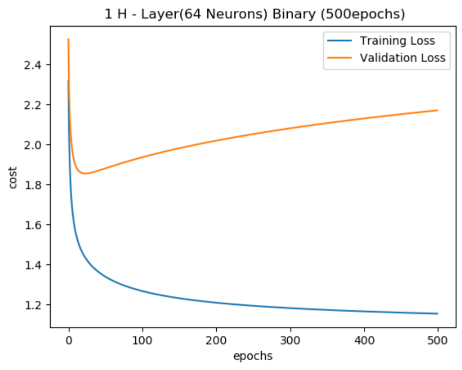

Fig. 7. Training and Validation losses for model with one hidden layer consisting of 64 neurons (Digit 0 vs ALL)

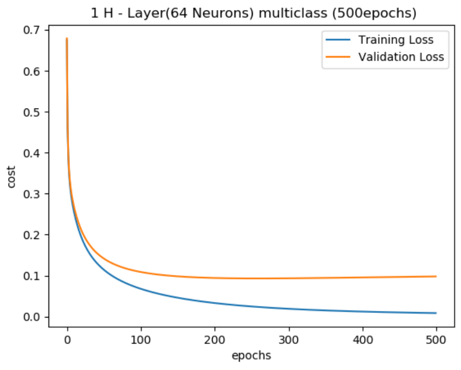

Fig. 9. Training and Validation losses for multiclass classification model

can see that the accuracy for classifying the digit 7 is less.

B. Model with one hidden layer

In this part of the experiment we have built a network with one hidden network consisting of 64 neurons. This network is also built for binary classification. Hyper parameters are as follows ***Learning rate : 0.1, Momentum : 0.9, Batch Size : 128, Epoch : 500***

So as previously all the labels of the digit 0 (zero) have been encoded by label 1 and rest of the digits are encoded to label 0.Now we train our model for 500 epoch and accumulate the training and validation losses.

In this case we have complex model than before since now we have 784 features in input layer , 64 neurons at hidden layer and 1 neurons at output layer. Fig. 7 shows the training and validation loss for the model. if we look at the figure it can be seen that, the validation loss is decreasing as the training loss until around 30 epoch and then the validation loss goes up massively. As our previous model this model also overfitting due to over training the model considering the complexity of the data. Another point here which may be worth mentioning is that, if we compare the fig. 5 and fig. **??** we see a that later is over-fitted more severely. The reason can be that later model has more capacity so it quickly learned the pattern and started to over fit more heavily. The overall accuracy of this model on test data is

Let us now do the same experiment that we did with our previous model. Again we will use the digit 7 instead of 0 for our binary classification. We can see the losses in the fig. 8 that, The model is not quite overfitting but the overall losses compared to before has greatly increased. This is consistent with observation that we had in the subsection VI-A. Also if we look closely to the curve it appears that we have stopped the training prematurely because the losses going down , it could have trained the model further to get better results.

Now if we compare the accuracy for both case in test data ,model for digit zero recognition has 99.17 percent accuracy and model for digit seven recognition has 98.24 percent accuracy. Which is almost 1 percent lower for the later model. Hence, we can say some digits are harder than other digits to recognize.

C. Model with one hidden layer for multiclass

Here we adopt the previous model for multiclass classification.In this model we have 784 input features , 64 units in hidden layer and 10 unit in output layer.The number of neuron is 10 in the output layer because we have 10 different

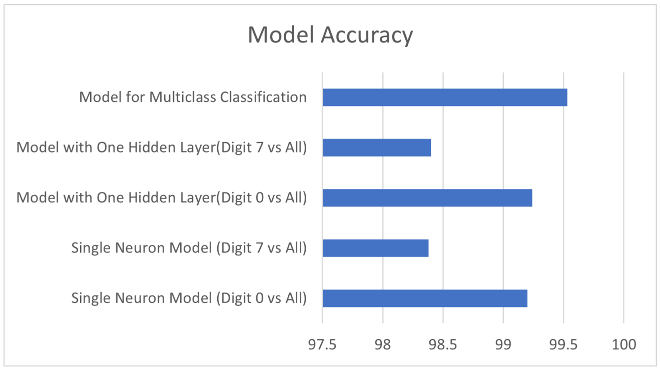

Fig. 10. Model accuracy on test data

digit to recognize. Also for this multi class problem we have used softmax activation function instead of sigmoid function. Although it is worth mentioning that, softmax function used for a binary classification problem is equivalent to a sigmoid activation function. . Fig. 1 shows the network architecture for multiclass classification.

Hyper parameters setting for the network are shown below ***Learning rate : 0.1, Momentum : 0.8, Batch Size : 128, Epoch : 500***

Losses are shown in fig. 9 for multiclass classification problem. In this case the model seems trained properly since the loss has gone down to almost 0. Also there not much deviation between the training loss and validation loss. The model has over all accuracy of 99.53 percent on our test set.

When we compare this multi class classification model with binary classification model from subsection VI-B with same hyper parameter setting surprisingly the over all loss(see fig. 7, 8 & 9) and accuracy (see fig. 10) for the multiclass is higher than binary classification.

Accuracy for all the variation of the model that we have discussed so far are shown in the fig.10. It can be clearly seen that multiclass classification has higher accuracy than any other category. Also it is noticeable that, in general model had difficulty in classifying the digit 7 rather than 0.

VII. Conclusions

In conclusion , it can observed that classification complexity or level of difficulty is not same for all the data even if they come from the same domain or distribution. As we have seen over training often leads over-fitted models hence carefully selection of the number of epoch could result in big performance boost on unseen data. Learning rate of the model is also another important parameter that can lead to a good performance of the model. Also momentum can be adjusted to avoid stucking in a local minima during the learning process.

References

[1] LeCun, Y. Cortes, C. (2010). MNIST handwritten digit database.

[2] Alzahrani, R., Parker, A.C. (2020). Neuromorphic Circuits With Neural Modulation Enhancing the Information Content of Neural Signaling. International Conference on Neuromorphic Systems 2020.

[3] https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/ accssed at 11:10 pm 2 Octobar 2020.

[4] Lehmann, E. L.; Casella, George (1998). Theory of Point Estimation (2nd ed.). New York: Springer. ISBN 978-0-387-98502-2. MR 1639875

[5] Glorot, X., Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256).