Project 1.1: Software 1.0 Versus Software 2.0

Baskar Adyar Krishnan
UBIT: BASKARAD
Person Number: 50291475

Abstract

The project is to compare two problem-solving approaches to software development: the logic-based approach (Software 1.0) and the machine learning approach (Software 2.0). We consider the task of the FizzBuzz problem. In this problem an integer divisible by 3 is printed as Fizz, an integer divisible by 5 is printed as Buzz. An integer divisible by both 3 and 5 is printed as FizzBuzz. If an integer is not divisible by 3 or 5 or 15, is printed as Other. Firstly, a python code is engaged to implement the traditional logic-based approach. Training and Testing Data for integers from 101 to 1000 and 1 to 100 respectively is generated using the simple python code. A Machine Learning Model is generated to implement the Software 2.0 approach. Various hyper-parameters required for the model is changed and the variations in the accuracy are of the model is analyzed.

1 Introduction

In this project, we are using one input layer. We have 900 numbers in the training set if we have to represent those numbers we need 10 bits in the binary form 2 power 10 = 1024. So 10 bits is enough to represent till 1024 numbers. Hence we are taking 10 neurons in the input layer. In the hidden layer, we are having 256 neurons. The number of neurons and number of layers in the hidden layers are also hyper-parameters and can be experimented by varying it.

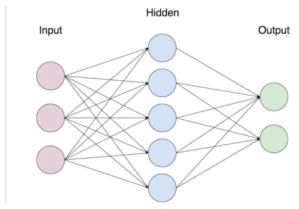

Figure 1: Sample Neural Network model

In this project, we are doing a classification problem. There are four possible output classes - Fizz, Buzz, FizzBuzz, Other. Hence, we need 4 neurons in the output layer.

2 Analysis

The objective is to change few hyper-parameters and analyze how the accuracy of the model is reacting to the changes in the hyper-parameters. Keras library in python is used for implementing the above network. Since this problem is quite a simple one we are going for only one layer between the input and the output later. Also, since there is no need to re-use any of the layers, we are using a sequential model.

Throughout this analysis, we are using softmax as the activation function to the weights between the hidden layer and the output. The reason for choosing softmax over other activation functions for this layer is, it gives the probabilities of different classes when the output is passed through this.Since we are using 4 output for classification, Softmax gives the probabilities of each.

The same loss function - categorical crossEntropy being used throughout this analysis. It is a loss function that is ideal for classification type problems with more than two classes.

20 percent of the training data is used as validation data set. While the model is learning, it simultaneously validates the model with that validation data set. Four variables are plotted against epochs to study the performance of the model in learning - Accuracy, Loss, Validation Accuracy and Validation Loss.

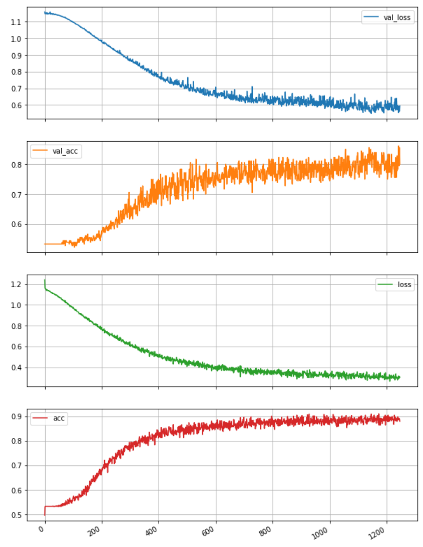

Figure 2:

2.1 With Dropouts

Dropout is a regularization technique which helps to avoid over-fitting by randomly dropping few nodes. Using “dropout", we randomly deactivate certain neurons in a layer. So, if 0.5 is set as the dropout, half of the activations of a layer is set to zero. This yet another hyper-parameter to be taken care of. Below is the graph showing the variations in the accuracy with respect to the changes in dropouts. Since higher dropout rate means a higher percentage of neurons getting deactivated in the

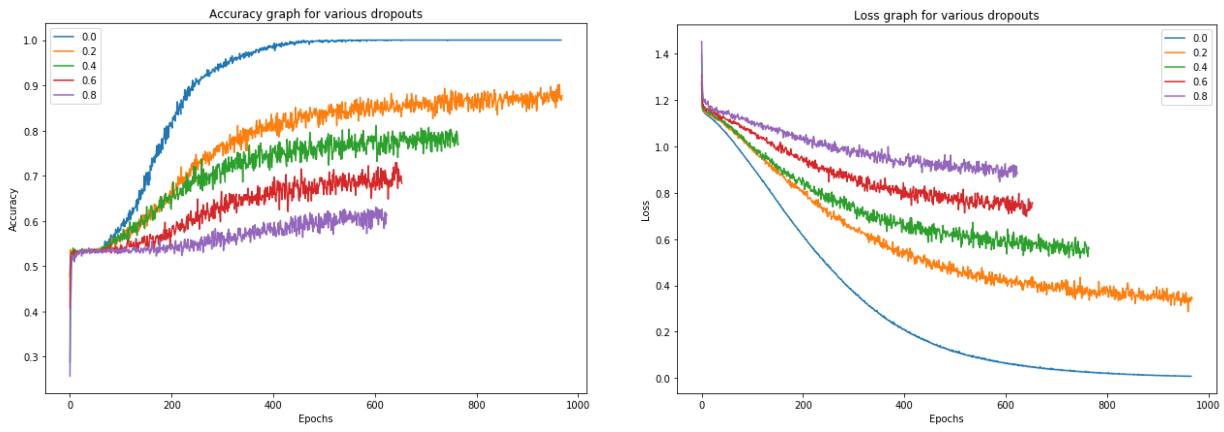

Figure 3: For Dropouts = 0.0, 0.2, 0.4, 0.6, 0.8

layer, the model would not learn anything from those deactivated inputs. Eventually, the chances of getting lower accuracy are possible. Figure 3 illustrates the change in accuracy with the change in dropouts.

2.2 With Activation functions

Why do we need activation function? They introduce nonlinear properties (non - linearities) to our network. Linear functions are always just a polynomial of one degree. If we plot these functions we would always get straight/rigid lines or planes. In order to get the curves(Non - linearities), we use activation function. Without an activation function, no matter how many layers of neural network we have, we would get a linear function. One more advantage is that we would want our function to be differentiable to do back propagation optimization strategy. Activation function helps us to get a differentiable function.

3 most popular activation function: sigmoid tanh relu. These three activation function are used and accuracy obtained along with the epochs is plotted in the Figure 4.

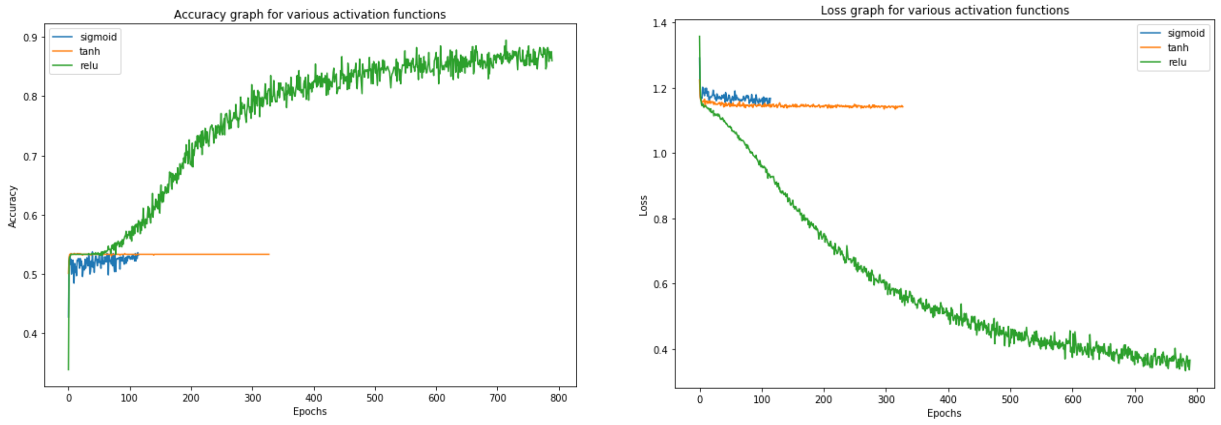

Figure 4: For different activation functions

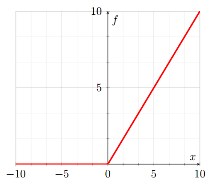

Figure 5: Relu function

As the graph illustrates clearly that Relu outperforms the other two activation functions. One major benefit of Relu is the reduced likelihood of the gradient to vanish. From the function graph of Relu in Figure 5, we can clearly see that Relu gives the same output when the function is greater than zero. In this regime, the gradient has a constant value. In contrast, the gradient of sigmoids becomes increasingly small as the absolute value of x increases. The constant gradient of Relu results in faster learning.

2.3 With Optimizers

Optimizer helps us to minimize an Error function. Here, the error function is dependent on input and the weights. Thus, optimizers not only help to minimize the Loss by the network's training process and also play a major role in the training process of the Models. Analysis of 5 different optimizers - *rmsprop, Adam, Adagrad, SGD, Adadelta* is shown in Figure 6. Out of all the optimizers analyzed for this FizzBuzz problem, Adam optimizer has performed well compared to others in learning. Adam stands for Adaptive Moment Estimation. Adam computes adaptive learning rates for each parameter. That is the reason why Adam's learning speed of the Model is quiet Fast and efficient compared to

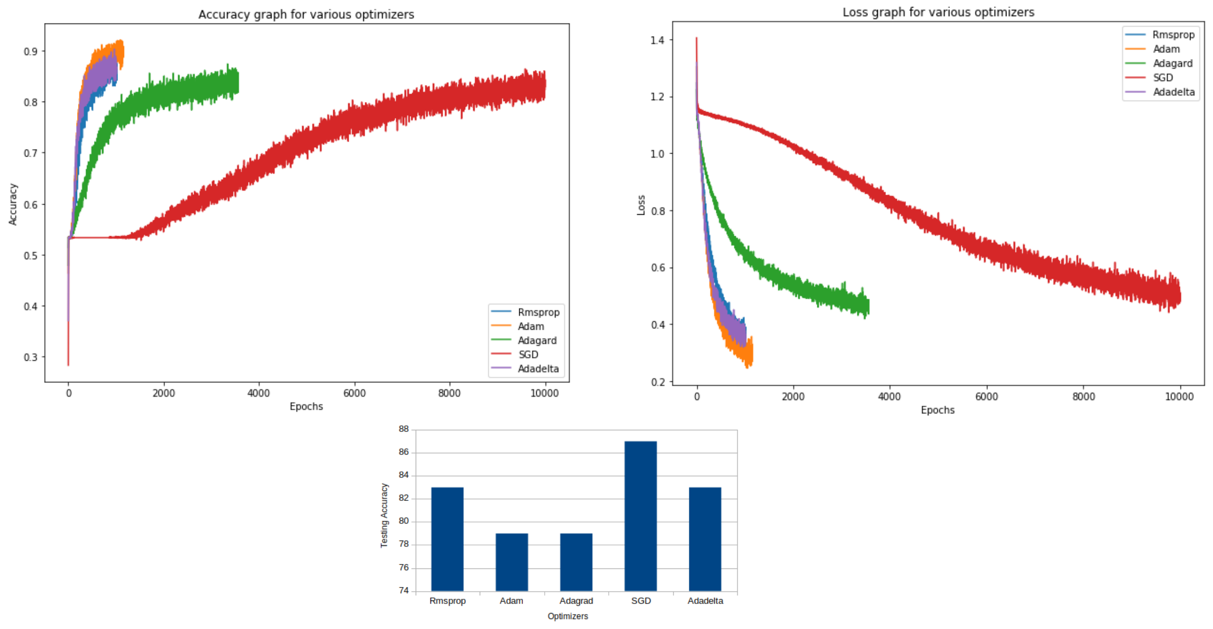

Figure 6: For different Optimizer functions

others. However, approximately Rmsprop and adadelta have performed almost similar to adam's performance during training for this particular problem of FizzBuzz. While looking at the testing accuracy results, rmsprop and adadelta give a better testing accuracy in some cases.

2.4 With Number of nodes in the hidden layer

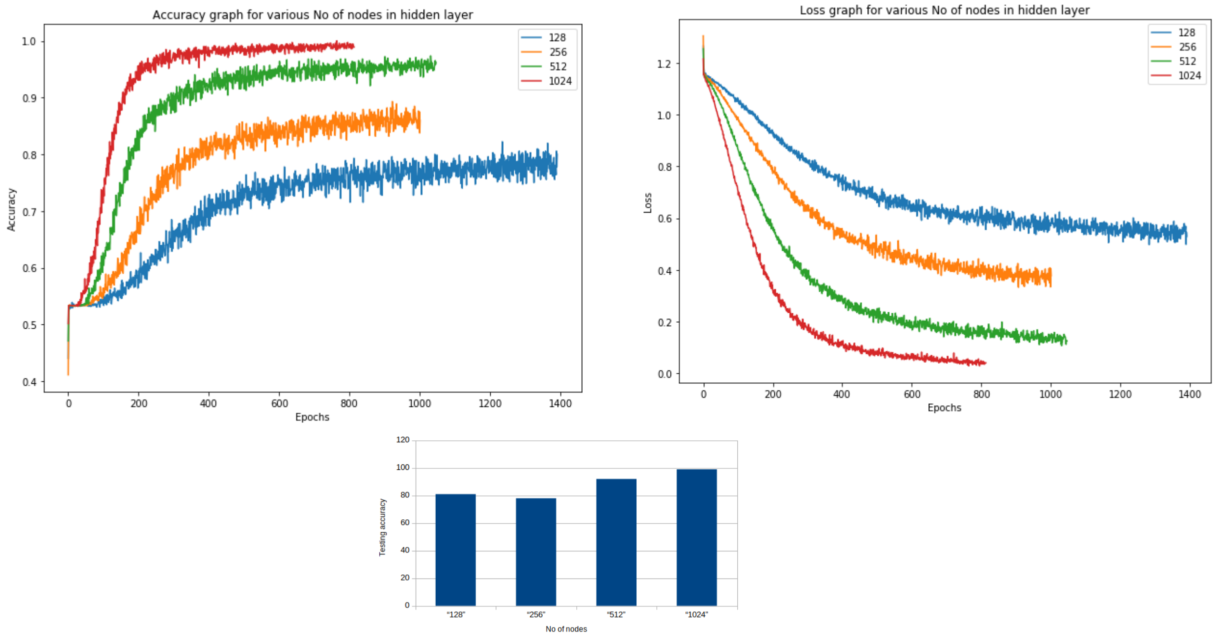

Figure 7: For different Optimizer functions

In this case, the number of nodes in the hidden layer is changed and the consequences of the changes in the model are observed. It is noted from the graphs that the higher the number of nodes in the hidden layer provided a higher learning rate and higher accuracy. With a large number of nodes in the hidden layer, the system has the possibility to learn more about the training data than what is required, we have to be aware of the case of over-fitting as well. So, it is recommended not to go for a very large number of nodes in the hidden layer. Similarly, on choosing a small number of nodes in the layers might have the negative effects like under-fitting and poor accuracy. Thus, the number of nodes in the layer must be tuned wisely based on the problem and training data set.

3 Summary

From above all observations and analysis we can figure out how a combination of hyper-parameters discussed above could possibly affect the Model's learning rate, loss, and accuracy. However, the same hyper-parameters may not work as expected for problems other than this FizzBuzz. Further observations are required for other hyper-parameters that is not considered in this report.

References

[1] https://stats.stackexchange.com/questions/126238/what-are-the-advantages-of-relu-over-sigmoid-function-in-deep-neural-networks.

[2] https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f