Project 1.1 – Report

MRUDULA Y

UB Person No: 50290843

I. PROBLEM STATEMENT

The task is to implement a program called "FizzBuzz" in two different ways – Software 1.0 and Software 2.0.

WHAT IS SOFTWARE 1.0 AND SOFTWARE 2.0 ?

Andrej Karpathy of Tesla came up with the concept of *Software 2.0*. Software 1.0 and Software 2.0 are both methods of implementing or coding a program to solve the given problem. Software 1.0 is the classic way programs are implemented in high level languages like C++, Python, Java etc where we give line by line instructions to the computer telling it what to do. The logic of the solution is coded by the programmer in Software 1.0. Whereas, in Software 2.0 the programmer doesn't code the logic but writes the program in an abstract manner providing the machine (computer) with the goal that has to be reached (Eg. Predicting the possibility of cancer in a patient), the computer utilizes its computational power and other resources (Eg. Input datasets, features) to come up with the required results. [1]

II. SOFTWARE 1.0 IMPLEMENTATION

As the "FizzBuzz" problem is a basic arithmetic problem, it was implemented using the if then else statements and basic arithmetic like modulo in Python. For example, we want the output as "Fizz" if the input integer is divisible by 3 and so on.

WHAT IS THE ACCURACY ACHIEVED AND WHY ?

The **accuracy** achieved was **1 or 100%**. The reason why a perfect accuracy could be reached was that it was implemented using the same logic that needs to be used in order to get the required outputs. The actual arithmetic and logic behind solving the problem was given as instructions to the computer line by line and the computer just executed them and got the output (Software 1.0).

III. SOFTWARE 2.0 IMPLEMENTATION

We used Neural Networks (NN) machine learning algorithm to solve the "FizzBuzz" problem. Tensorflow library in Python was used in order to implement this algorithm. We give the input data (set of integers) and the output we want (goal [set of labels]) to the neural network and let the computer figure out the pattern (the logic) to get the required output (Software 2.0).

PARAMETERS OF NEURAL NETWORKS AND THEIR MEANING

- Number of Epochs – One epoch is the entire dataset. In order to train the neural network and make it learn a pattern we need to repeatedly feed the dataset to it. The number of times we feed the same dataset to the neural network is number of epochs. [2]
- Batch Size – Most of the times the dataset is too huge to be fed to the neural network all at once. So, the input dataset (epoch) is divided into several smaller batches. The number of data points that each batch has is the batch size. [2]

- Number of Hidden Layers – This is the layer of nodes between the input and the output layer. They take weighted inputs and produce an output through an activation function. [3]
- Activation functions – Activation functions are mathematical functions that are used to make sense out of the output given by the nodes by taking several weighted inputs. For example – In the case of "FizzBuzz" we need the output to be any of the four – Fizz, Buzz, FizzBuzz and Other. Now, after so much of computation on the input integers we will have complex numbers. On applying an appropriate activation function we can transform the output complex numbers in such a way that they lie within a certain range. Then we can define different thresholds for each of the four outputs. Depending on the output of the activation number we can classify them under four labels.
- Loss function – This is used to measure the inconsistency between predicted value and the actual label. The lesser the value of the loss function the better our model is. [4]
- Optimization methods – This is nothing but the algorithms used to optimize the results given by the neural network model. It is used to decrease the value of loss function and increase accuracy.
- Dropout rates – The set of data that is dropped out or ignored during the training phase of a neural network to reduce the chance of overfitting. [4]
- Distribution used for weights – The weights to the inputs are assigned at random but based on some probability distribution like normal, uniform etc.
- Learning Rate – This is a parameter of Gradient descent (minima of a curve). Gradient Descent is a iterative algorithm to make the results most optimal. So bigger size steps will be taken initially to reach the minima and as the point on graph goes down the size of step becomes smaller. This step is learning rate.

A. INITIAL STATE

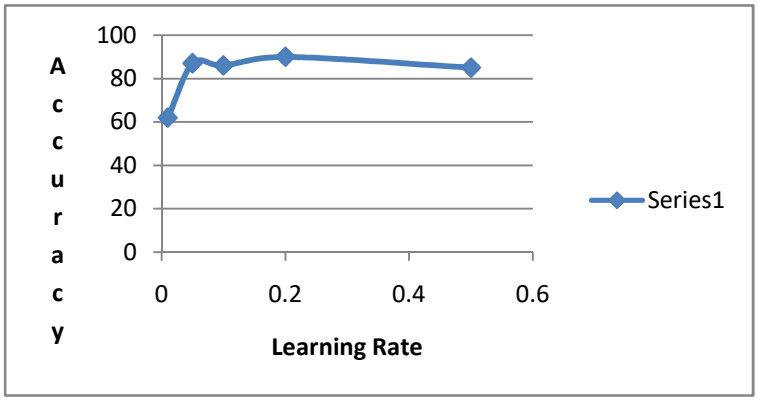
Initially, the given program was using the following hyper-parameters – Learning rate = 0.1; Number of epochs = 5000; Batch size = 128; Number of hidden neurons in layer 1 = 150; Activation function = ReLu; Optimizer = Gradient Descent Optimizer; Distribution of weight = random normal;

The initial accuracy obtained is: 86%.

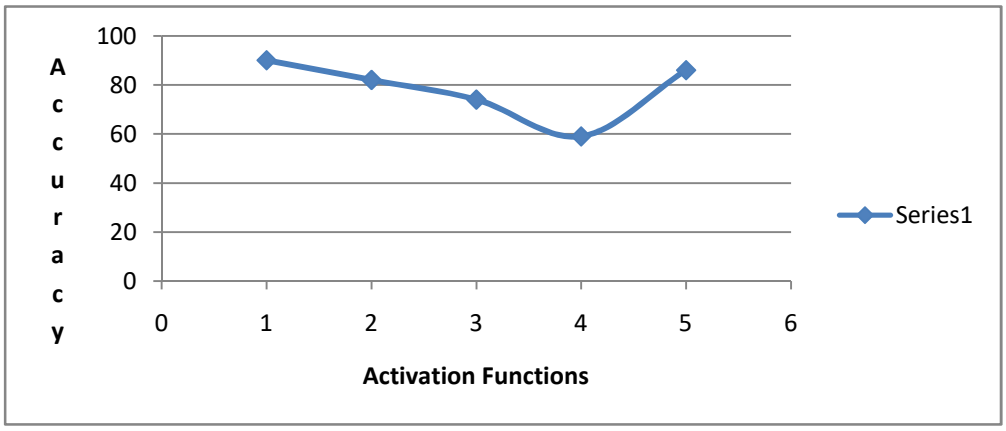
B. EXPERIMENTATION

Hyper-parameters such as Learning rate, Number of nodes in layer 1, number of epochs, batch size, activation function and optimization methods were tweaked while keeping others constant. The parameter that gave the best accuracy while analyzing one parameter, was kept when the next hyper-parameter was analyzed in the hope of reaching the best parameter combination over the course of experimentation.

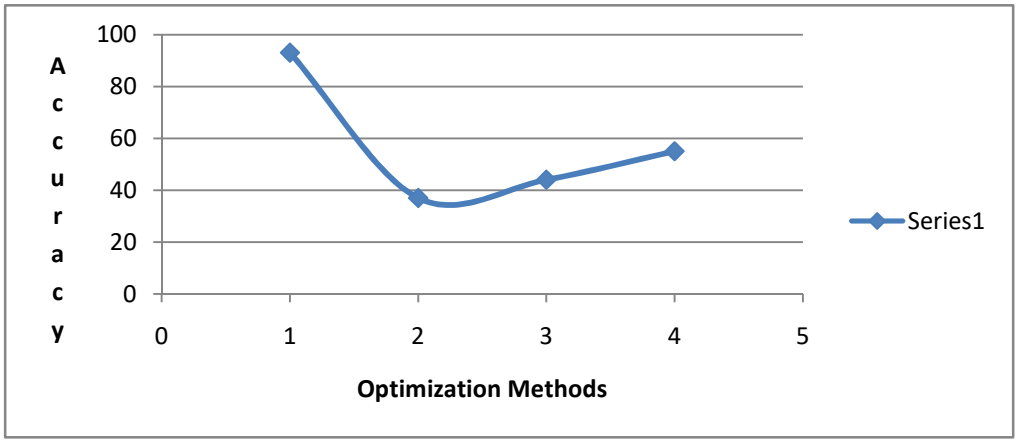
The graphs below show how various parameters vary the accuracy of the model.



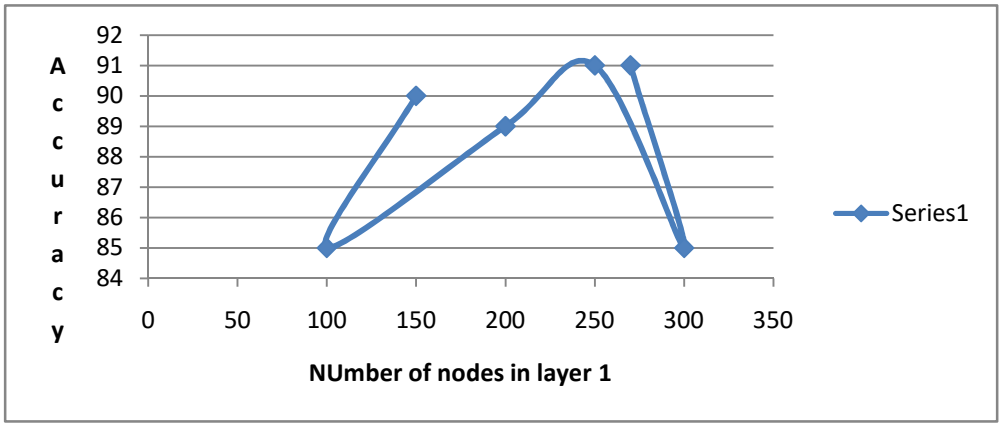
(a)



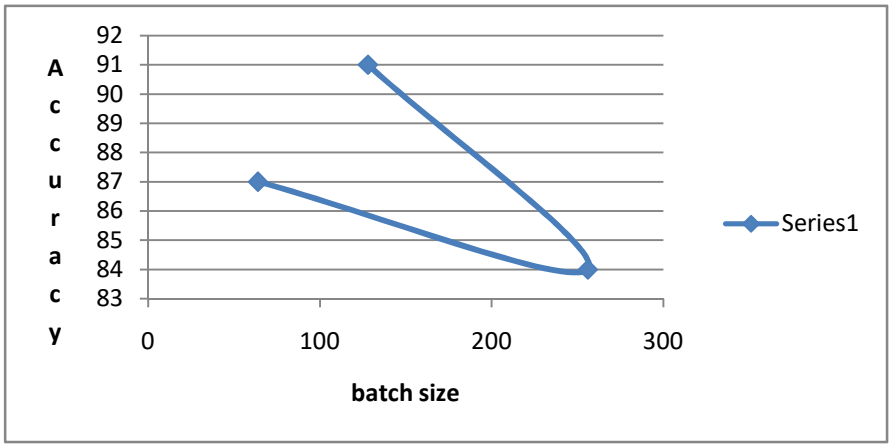
(b) 1: relu 2: tanh 3: sigmoid 4: softmax 5: leaky_relu activation functions.



(c) 1: Gradient Descent Optimizer 2: Adadelta optimizer 3: Adam optimizer 4: RMSProp



(d)



(e)

C. ANALYSIS AND RESULTS

LEARNING RATE – A too small learning rate causes low accuracy, the accuracy does increase till a learning rate of 0.2, after that it again starts decreasing. This shows that neither too low nor too high learning rates are recommended. These should be optimal.

BATCH SIZE – This is again a parameter that needs to be medium and optimal. Nor to high nor too low. This means if too much data is fed the model is unable to process it and if too less data is fed at a time it gives no information to the model to learn.

NUMBER OF NODES IN LAYER 1 – We can increase the number of nodes to a certain extent as the accuracy grows with it but after 250 the accuracy started decreasing. The more the iterations using more nodes, the model understands the pattern better but if the nodes are too high overfitting will occur hence decreasing the overall accuracy.

ACTIVATION FUNCTION – ReLu gives the best accuracy among all of them. In ReLU, the data points below zero are nullified and the data points above zero are taken as it is. In FizzBuzz, the data set is a set of integers which have to be used as it is in a modulo function to know divisibility. In ReLu the integers do not change and hence stay very near to the software 1.0 logic. Hence ReLu works well in FixxBuzz case.

OPTIMIZATION METHODS – The Gradient Descent Optimizer works best.

The final accuracy = 93%
Hyper-parameters are as follows :

Learning Rate: 0.2
Gradient Descent Optimizer
ReLu Activation Function
Number of Epochs: 5000
Batch size: 128
Number of nodes in layer 1: 250

IV. CONCLUSION

Software 1.0 is better for this problem than Software 2.0. This problem being a simple and straightforward using arithmetic logic, Software 1.0 works well and gives 100% accuracy. But, training the neural network in a way that the model gets the pattern and gives 100% accuracy is time taking and lot of work (adjusting hyper-parameters appropriately). Using Software 2.0 implementation 93% accuracy for the overall model was obtained.

Although, I feel Software 2.0 is best for complex problems, might be the only way to get some solution in some cases.

V. REFERENCES

[1] Software 2.0 – Andrej Karpathy – Medium [https://medium.com]
[2] www.towardsdatascience .com
[3] https://www.techopedia.com
[4] https://isaacchanghau.github.io
[5] Learning Less to Learn Better – Droput in (Deep) Machine Learning – Medium [https://medium.com]
[6]