CSE574 Introducion to Machine Learning

Prof: Sargur Srihari

Project 1.1Fizzbuzz Report.

Submitted by
Sai Varun Alapati
Ubit : saivarun
Personal Number :50290571

Contents

1. Introduction

This project is to compare two Fizz Buzz problem solving approaches to software development: the logic-based approach (Software 1.0) and the machine learning approach (Software 2.0).

2. Problem definition

The problem is Fizz Buzz, In this problem an integer divisible by 3 is printed as *Fizz*, and integer divisible by 5 is printed as *Buzz*. An integer divisible by both 3 and 5 is printed as *Fizz Buzz*. If an integer is not divisible by 3 or 5 or 15, it simply prints the input as output (for this last case, the input number can be classified as other in Software 2.0,)

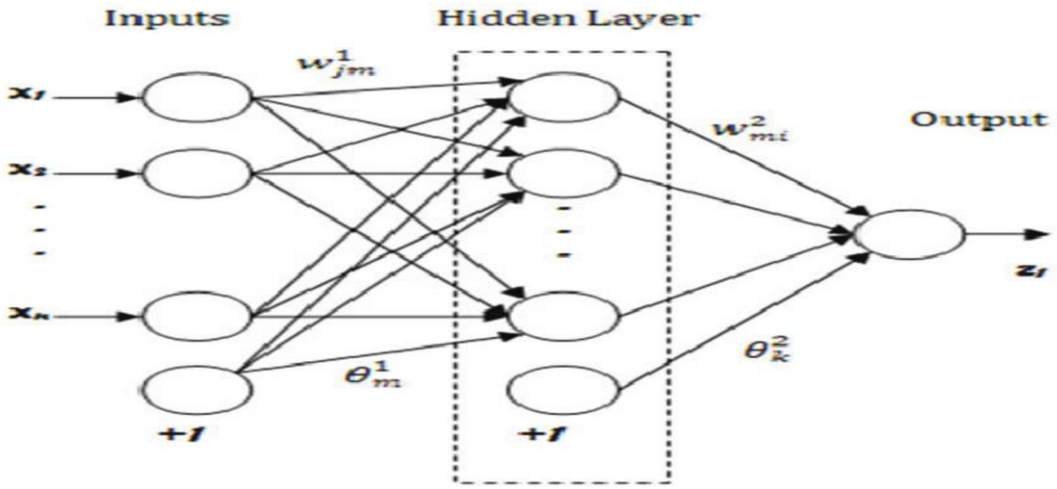
3. Software 1.0 :

We have written the logic implemented the Fizz buzz program in the python code in traditional way

4. Software 2.0 :

a. The learning system

Making a model helps to explain a system and to study the effects of different components, and to make predictions about behavior. The model is created in keras which uses high level API built on Tensor Flow. It is more user-friendly and easy to use as compared to Tensor flow**.** Using Sequential model which is just a linear stack of layers. The Artificial Neural Network model will have inputs which are 10 , one dense hidden layer consists of several nodes, activation function and 2nd layer as outputs as shown in below figure.



The activation function introduce non linear properties to the network and converts input signal of a node to an output signal which can be used as a input to the next dense layer. Later To prevent Neural network from overfitting we are adding dropouts to the hidden layer which drops random weights which are causing overfitting to the model. To reduce the influence of extreme values in the data without removing them from the data set we are adding softmax activation function. Later Categorical crossentropy Loss function is used to calculate the amount of inaccuracy.

5. Experimental Values :

a. Hyper parameters Used:

The following are the hyper parameters used in this model.

- Number of Neuron/Nodes : These are the number of nodes/neurons in the first dense layer.
- Droupouts : This is the percentage of dropout rate Activation function
- Optimiser : This the function name of the optimizer used
- Number of hidden layers : These are number of hidden layers in the overall neural network used.
- Input batch size : This the number of data samples sent into model per single batch
- Number of epochs :This is the number of iterations the model should run for the given data set.
- Validation data split – This is the percentage of the data samples used for validating the model in each epoch.

b. Varying Number of Neurons/Nodes :

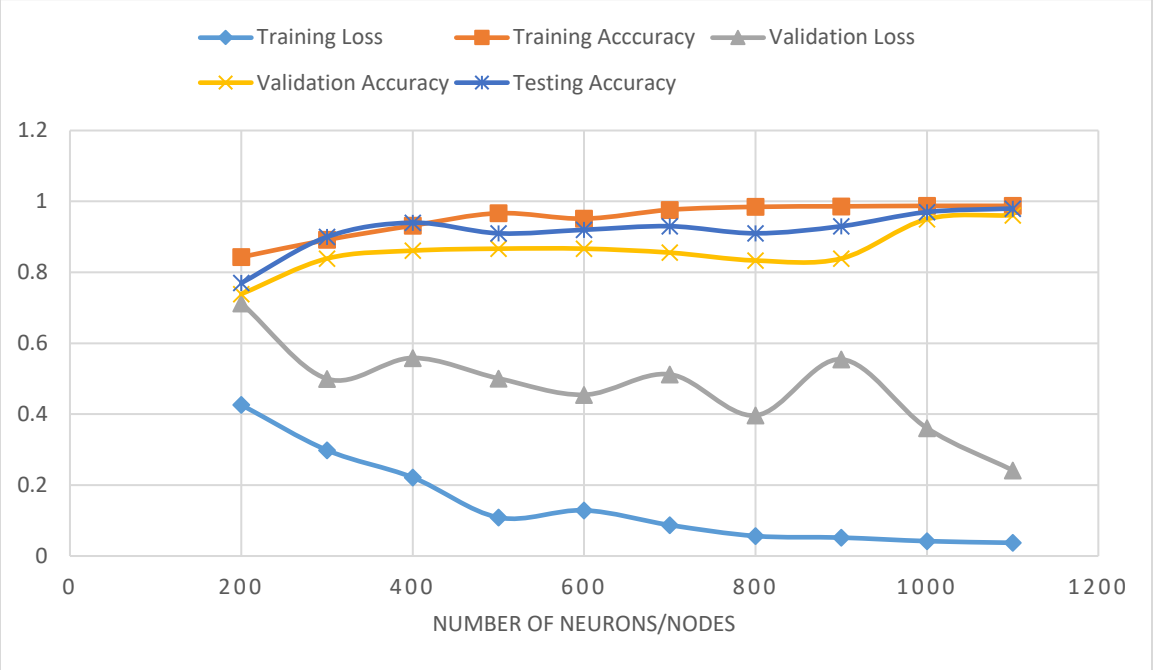
These are number of hidden layers in the overall neural network used. The below hyperparameters are fixed and nodes are varied from 100 to 1100 in the step of 100.

Hyper Parameters	Value
Droupouts	0.2
Activation function	Relu
Optimiser	Rmsprop
Number of hidden layers	1
Input batch size	128
Number of epochs	10000
Validation data split –	0.2

The below table gives the experimental accuracy measures with respect to the number of neurons varied.

Number of Neurons/Nodes	Training Loss	Training Accuracy	Validation loss	Validation Accuracy	Testing Accuracy
100	0.6864	0.7250	1.0636	0.5667	0.69
200	0.4262	0.8431	0.7128	0.7389	0.77
300	0.2984	0.8917	0.4997	0.8389	0.90

400	0.2215	0.9319	0.5586	0.8611	0.94
500	0.1087	0.9667	0.5009	0.8667	0.91
600	0.1289	0.9514	0.4547	0.8667	0.92
700	0.0873	0.9764	0.5122	0.8556	0.93
800	0.0564	0.9847	0.3965	0.8333	0.91
900	0.0519	0.9861	0.5547	0.8389	0.93
1000	0.0419	0.9875	0.3607	0.9500	0.97
1100	0.0373	0.9875	0.2419	0.9611	0.98



The above graph interprets the accuracy measures with the number of nodes. From the above experiment we can see the number of neurons does effect the model performance. Here wh en the neural network has few neurons below 300 it didn't had the capacity to learn enough of the underlying patterns to distinguish numbers properly. Later when we increased the number of neur ons it does helped the model to learn from the data.

c. Varying Number of Dropouts

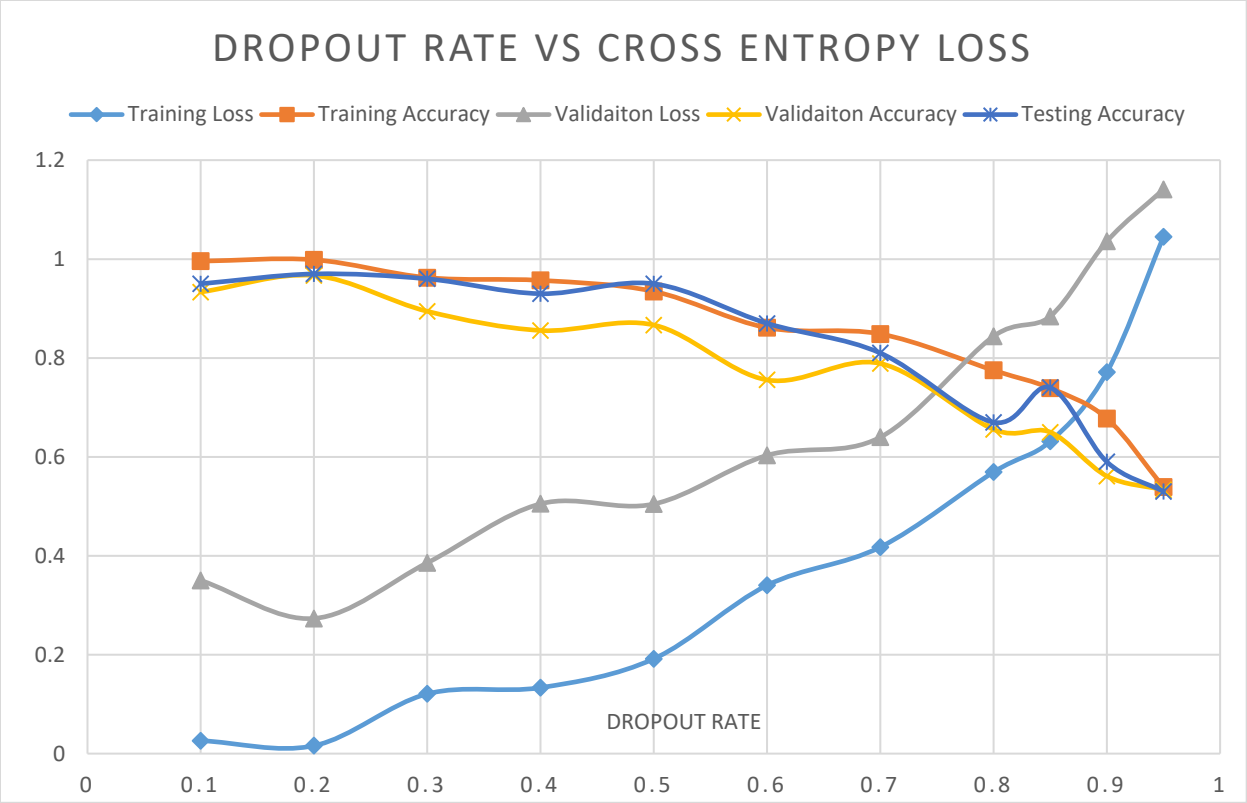
To prevent neural network from overfitting we use dropouts to the hidden layer which drops random weights which are causing overfitting to the model.

Hyper Parameters	Value
Number of Neurons	1024
Activation function	Relu
Optimiser	Rmsprop
Number of hidden layers	1
Input batch size	128
Number of epochs	10000

Validation data split –	0.2

The below are the experimental values of accuracy measures when dropouts are varied from 0.1 to 0.95

Dropouts	Training Loss	Training Accuracy	Validation loss	Validation Accuracy	Testing Accuracy
0.1	0.0262	0.9958	0.3502	0.9333	0.95
0.2	0.0160	0.9986	0.2733	0.9667	0.97
0.3	0.1208	0.9625	0.3858	0.8944	0.96
0.4	0.1334	0.9569	0.5051	0.8556	0.93
0.5	0.1917	0.9347	0.5047	0.8667	0.95
0.6	0.3403	0.8611	0.6031	0.7556	0.87
0.7	0.4176	0.8486	0.6397	0.7889	0.81
0.8	0.5693	0.7750	0.8439	0.6556	0.67
0.85	0.6314	0.7389	0.8841	0.6500	0.74
0.9	0.7711	0.6778	1.0359	0.5611	0.59
0.95	1.0450	0.5389	1.1408	0.5333	0.53



The above graph interprets the cross entropy loss of training and validation loss to the dropout rat e. So from the above experiments we conclude that with increasing in validation accuracy and de crease in loss initially before the trend to go down.

d. Varying Activation function - sigmoid, tanh, relu

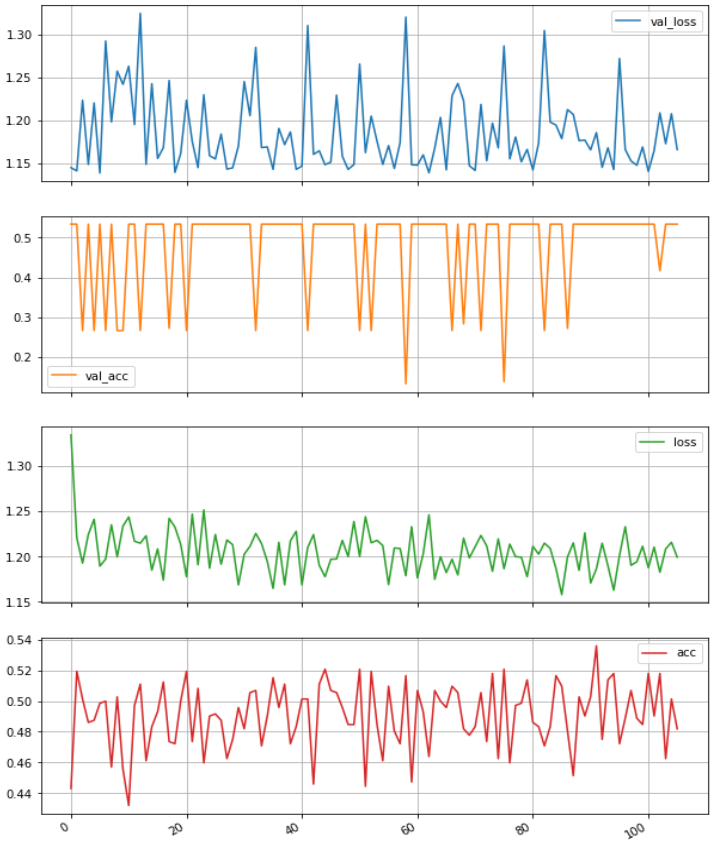
The activation function introduce non linear properties to the network and converts input signal of a node to an output signal which can be used as a input to the next dense layer. The below hyper parameters are fixed for varying the activation functions.

Number of Neuron	1024
Droupouts	0.2
Optimiser	Rmspprop
Number of hidden layers	1
Input batch size	128
Number of epochs	10000
Validation data split –	0.2

i. Sigmoid :

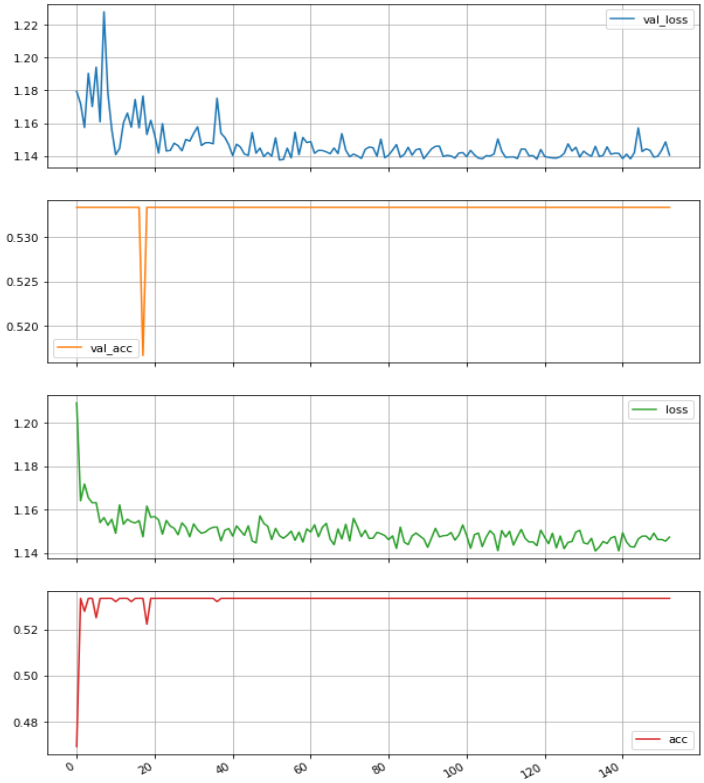
Sigmoid activation function of form f(x) = 1/1+exp and its range is between 0 and 1 and it has vanishing gradient problem. It also makes the gradient updates go too far in different directions. The accuracy measures are also not stable and which are shown in the below table and the graph with respect to epochs are shown.

Accuracy Measures	Value
Training Loss	1.1992
Training Accuracy	0.4819
Validation loss	1.1663
Validation Accuracy	0.5333
Testing Accuracy	0.53



ii. Tanh :

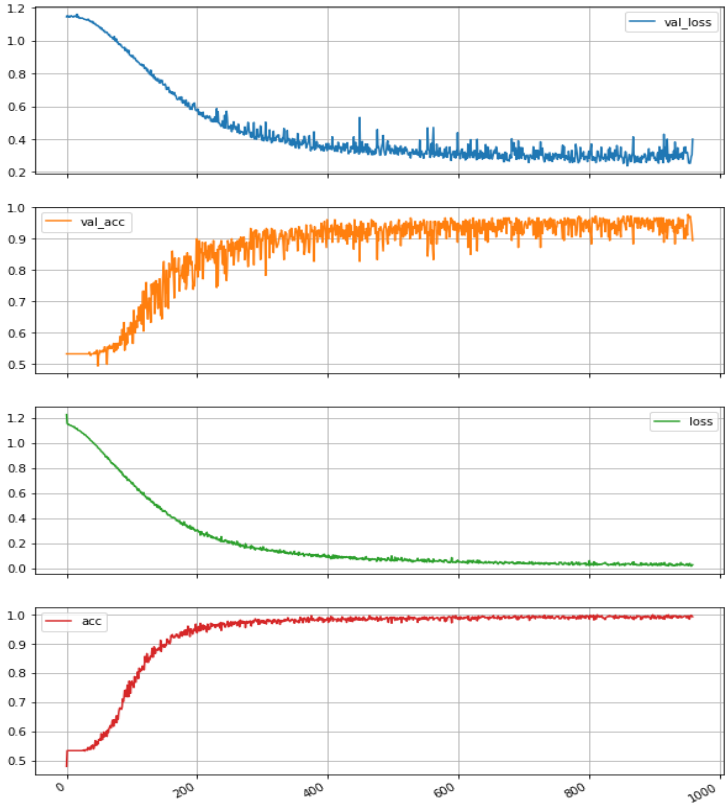
Tan's mathematical formula is **f(x) = 1—exp(-2x) / 1 + exp(-2x)** and its range is zero centered which lie between -1 to 1. Its optimization is simple and a lways preferred overed sigmoid. It also suffers fro m vanishing gradient problem. The accuracy meas ures are also not stable which are shown in the bel ow table and the graph with respect to epochs are s hown .



Accuracy Measures	Value
Training Loss	1.1473
Training Accuracy	0.5333
Validation loss	1.1405
Validation Accuracy	0.5333
Testing Accuracy	0.53

iii. **Relu :**

Relu's mathematical formula is *R(x) = max(0,x) i.e if x < 0 , R(x) = 0 and if x >= 0 , R(x) = x.* Its very easy and efficient which motivates deep learning models to use them over sigmoid and t anh. It also avoids vanishing gradient problem. The accuracy measures are good which are sho wn in the below table and the graph with respec t to epochs are shown .



Accuracy Measures	Value
Training Loss	0.0296
Training Accuracy	0.9931
Validation loss	0.4004
Validation Accuracy	0.8944
Testing Accuracy	0.95

e. Varying Optimiser - SGD, Adagrad, Adadelta, Rmsprop and Adam

Optimization algorithms helps to minimize/maximize the Error Funciton E(x) which are used in the calculation of output values (Y) from the predictors used in the model. In this model it minimizes the loss by the network's training process and also play a major role in the ***training*** process of the Neural Network Model . The below hyper parameters are fixed for varying the optimisers.

Number of Neuron	1024
Droupouts	0.2
Activation function	relu
Number of hidden layers	1
Input batch size	128
Number of epochs	10000
Validation data split –	0.2

i. Rmsprop

RMSProp is also a method in which the le arning rate is adapted for each of the para meters it divides the learning rate for a we ight by a running average of the magnitud es of recent gradients for that weight.

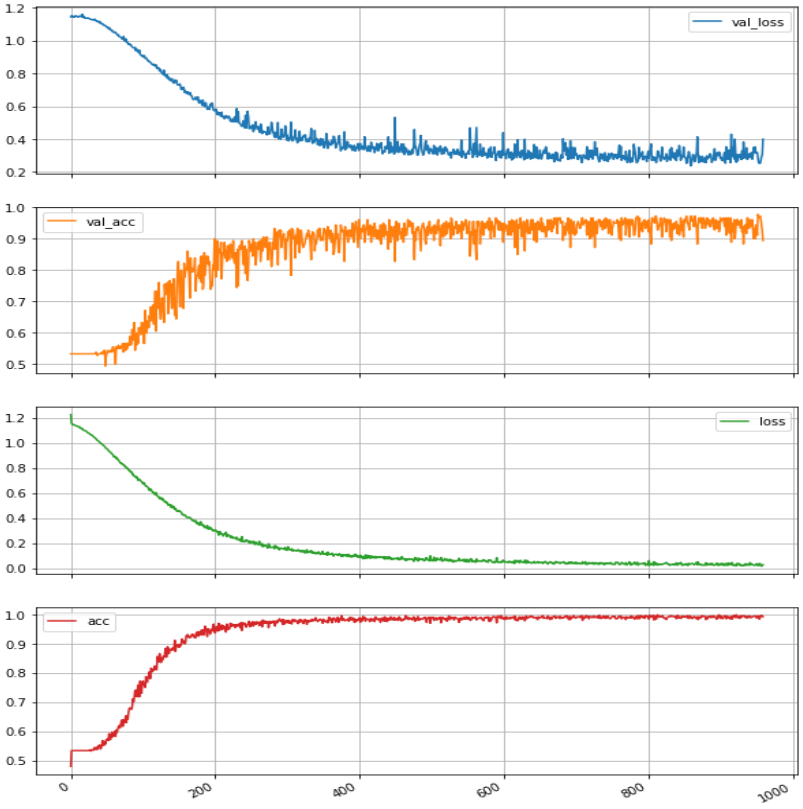
$$v(w,t) := \gamma v(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2$$

where, Y is the forgetting factor.
And the parameters are updated as,

RMSProp has shown excellent adaptation of learning rate in different applications.

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla Q_i(w)$$

The accuracy measures are good which ar e shown in the below table and the graph with respect to epochs are shown .

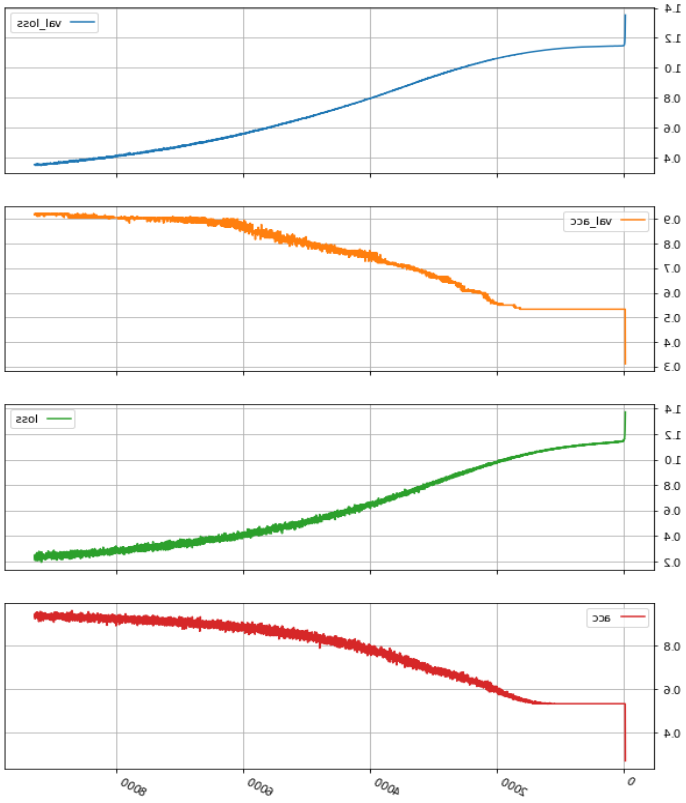


Accuracy Measures	Value
Training Loss	0.0296
Training Accuracy	0.9931
Validation loss	0.4004
Validation Accuracy	0.8944
Testing Accuracy	0.95

ii. SGD Stochastic :

SCD performs one update at a time.
$\theta=\theta-\eta\cdot\nabla J(\theta;x(i);y(i))$, where {x(i) ,y(i)} are the training examples.
Now due to these frequent updates ,parameters updates have high variance and causes the Loss function to fluctuate to different intensities.

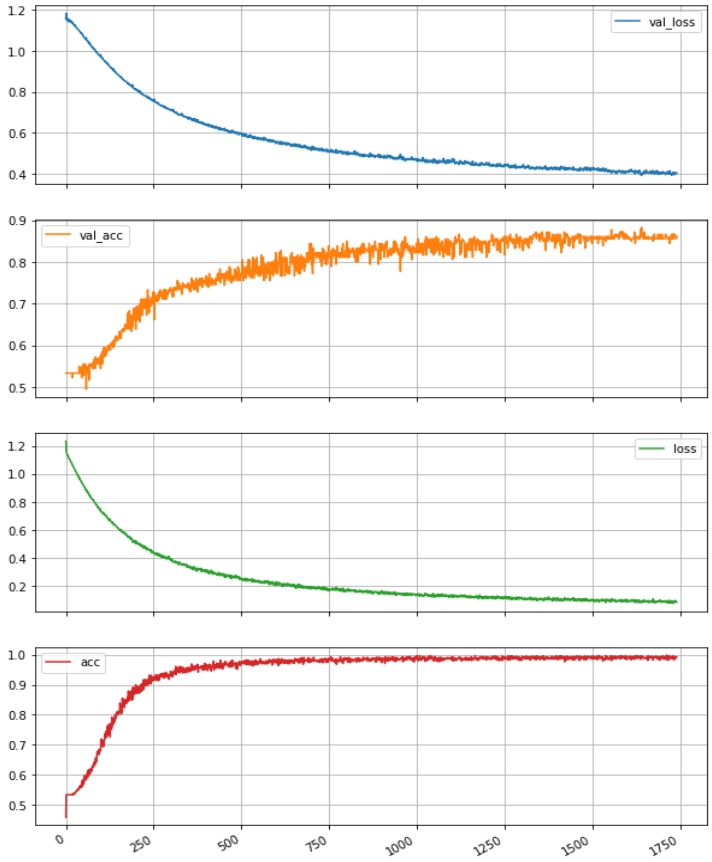
The accuracy measures are best which are shown in the below table and the graph with respect to epochs are shown .



Accuracy Measures	Value
Training Loss	0.2485
Training Accuracy	0.9250
Validation loss	0.3524
Validation Accuracy	0.9167
Testing Accuracy	97

iii. Adagrad

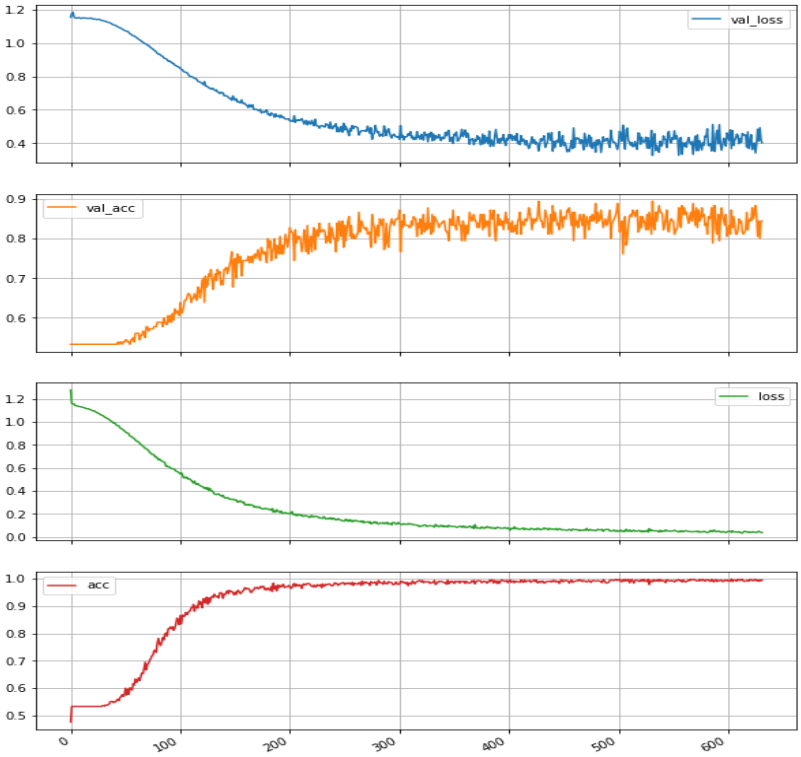
It simply allows the learning Rate -η to adapt based on the parameters. So it makes big updates for infrequent parameters and small updates for frequent parameters. For this reason, it is well-suited for dealing with sparse data.
It uses a different learning Rate for every parameter θ at a time step based on the past gradients which were computed for that parameter.
The accuracy measures are good which are shown in the below table and the graph with respect to epochs are shown.



Accuracy Measures	Value
Training Loss	0.0868
Training Accuracy	0.9917
Validation loss	0.4037
Validation Accuracy	0.8611
Testing Accuracy	0.97

iv. Adadelta

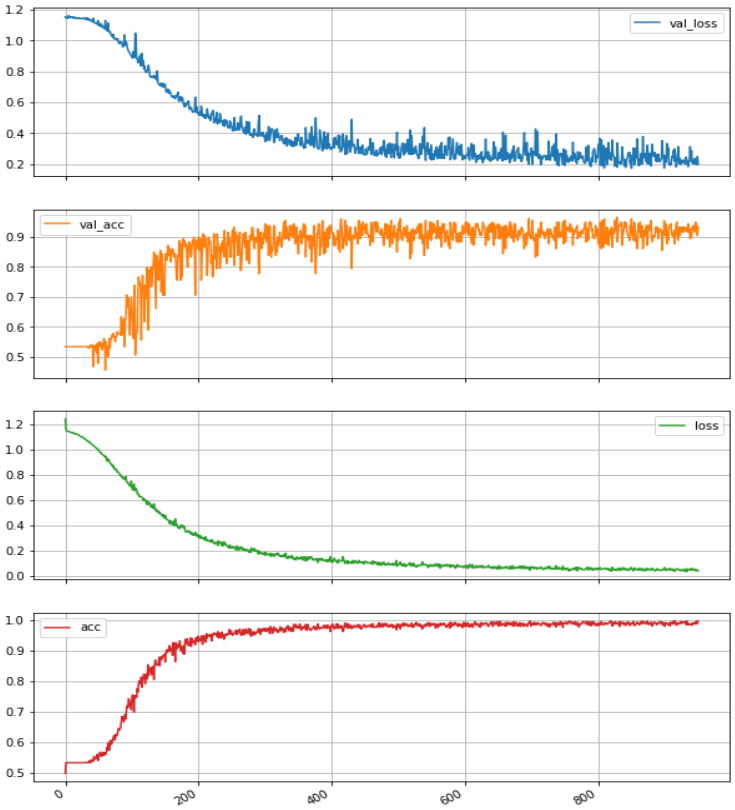
It is an extension of AdaGrad which tends to remove the decaying learning Rate problem of it. Instead of accumulating all previous squared gradients, Adadeltal imits the window of accumulated past gradients to some fixed size w.
The accuracy measures are best which are shown in the below table and the graph with respect to epochs are shown .



Accuracy Measures	Value
Training Loss	0.0357
Training Accuracy	0.9944
Validation loss	0.1990
Validation Accuracy	0.9333
Testing Accuracy	0.98

v. Adam

Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients like AdaDelta ,Adam also keeps an exponentially decaying average of past gradients M(t), similar to momentum
The accuracy measures are best which are shown in the below table and the graph with respect to epochs are shown .



Accuracy Measures	Value
Training Loss	0.0606
Training Accuracy	0.9861
Validation loss	0.3103
Validation Accuracy	0.8889
Testing Accuracy	0.98

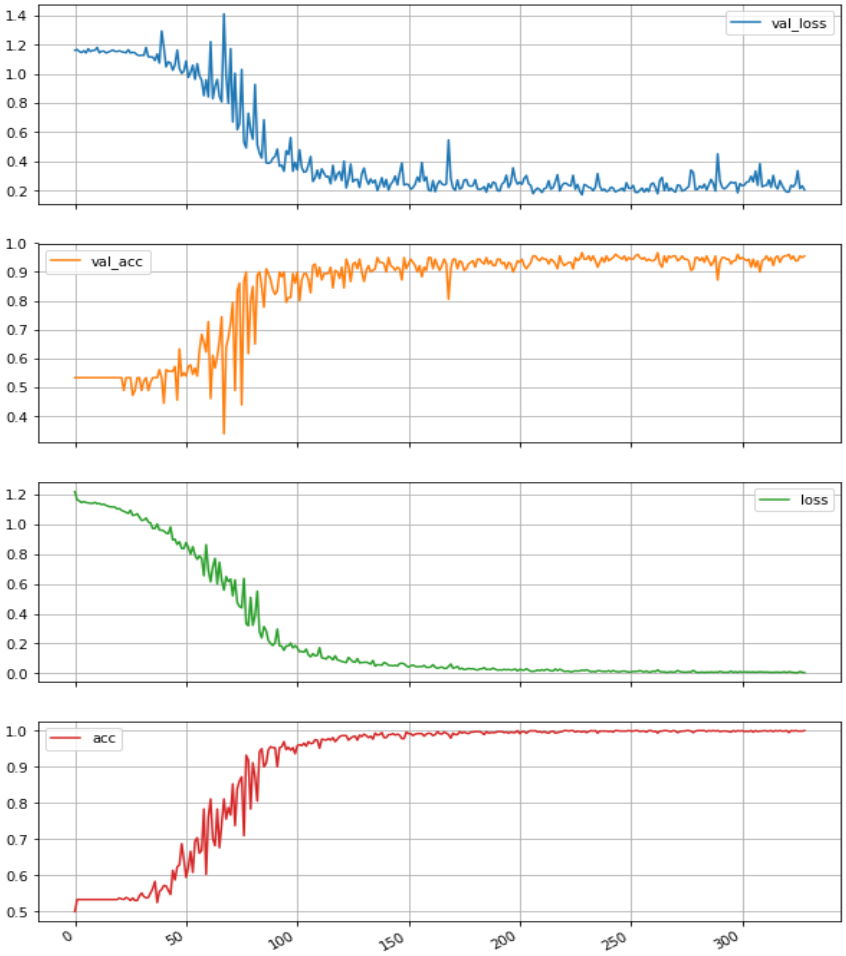
f. Adding a Hidden Layer :

Adding a Hidden layers can represent an arbitrary decision boundary to arbitrary accuracy with r ational activation functions and can approximate any smooth mapping to any accuracy. The belo w hyper parameters used while experimenting.

Number of Neuron	1024+1024
Droupouts	0.2
Activation function	Relu
Optimiser	Adadelta
Number of hidden layers	2
Input batch size	128
Number of epochs	10000
Validation data split –	0.2

Adding a hidden layer allowed simple r translations between levels of abstra ction.The accuracy measures are best which are shown in the below table an d the graph with respect to epochs are shown .

Accuracy Measures	Value
Training Loss	0.0065
Training Accuracy	1.0000
Validation loss	0.2046
Validation Accuracy	0.9556
Testing Accuracy	0.98



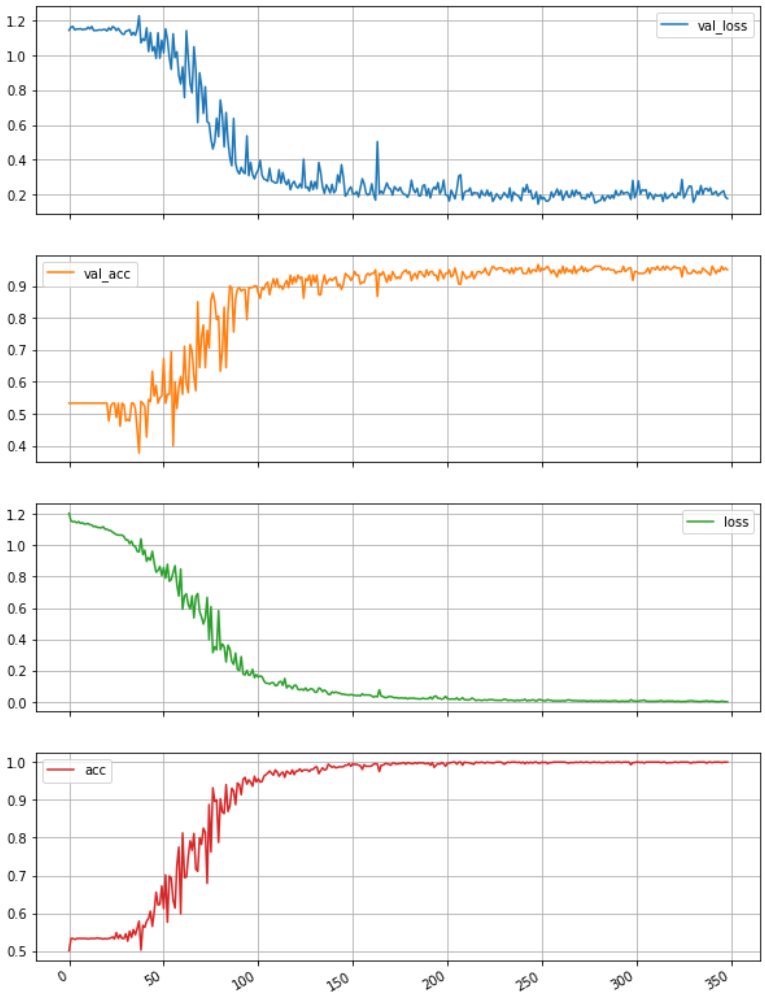
6. Creating a model

Based on the experimental results of varying hyper parameters we have chosen the following hyp er parameters.

Number of Neuron	1024+1024
Droupouts	0.2
Activation function	Relu
Optimiser	Adadelta
Number of hidden layers	2
Input batch size	128
Number of epochs	10000
Validation data split –	0.2

- Adding a additional layer gave smooth mapping to the accuracy and gave the best so used the 2nd hidden layer.
- We increased additional neurons which increases the the capacity to learn enough of the u nderlying patterns to distinguish numbers properly.
- From above dropout experiment we found the accuracy measures are increasing and decre ased significantly after 0.2.
- Since relu avoids vanishing gradient problem and is efficient which doesn't limit the valu es.
- Used Adadelta Optimiser which Preventing Vanishing(decaying) learning Rates.
  for each parameter and gave the better accuracy compared to others.

The accuracy measures are best which are shown in the below table and the graph with respect to e pochs are shown .



Accuracy Measures	Value
Training Loss	0.0028
Training Accuracy	1.0000
Validation loss	0.1757
Validation Accuracy	0.9500
Testing Accuracy	1.0000

The class wise accuracies are shown below.

Classwise Accuracy	Value
Fizz	1.0000
Buzz	1.0000
Fizzbuzz	1.0000
Other	1.0000
Total Accuracy	1.0000

7. Summary :

In This project, I brushed up my python and machine learning frameworks and understoo od the basic concepts of the Machine Learning, Tensorflow and keras. I tuned this model to the b est of the hyper parameters available and got accuracy.

8. References :

https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f
https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f
https://www.researchgate.net/profile/Andreas_Andreou4/publication/221911786/figure/fig1/AS:305296409939974@1449799745064/A-feed-forward-Multi-Layer-Perceptron-Neural-Network-Equation-9-specifies-how-the.png