Assignment 1
Deep Learning E0 250 (CSA)
The FizzBuzz Problem
Rahul John Roy (M.Tech CDS) - 16703

1. The Problem.

In this task 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 FizzBuzz. If the number is not divisible by both 3 and 5, then the number itself is printed as the output.

2. Structure.

Configuration used:

- Python 3.7.6
- Keras 2.3.1
- Tensorflow 2.0.0

The following files will be part of this assignment:

- main.py - The main file of this assignment, which invokes the saved model and tests the input from a txt file which can be passed by command line arguments. It generates two output files - 'Software1.txt' and 'Software2.txt' which are generated by the logic based and the machine learning code respectively.
- model.py - This is where the neural network is defined. The definitions were done using Keras using TF in the backend.
- model.h5 - This file is generated when the file 'model.py' is run, and this will subsequently get run from the 'main.py' code.

3. Methodology.

Standard Logic:
The divisibility with 3 and 5 is evaluated using the modulo operator and various conditional statements are evaluated and the corresponding output is printed onto the corresponding file 'Software1.txt'.

Machine Learning:
The is a classification problem and there are four classes involved (Fizz, Buzz, FizzBuzz, and the number itself). As per the problem statement, the training input should be from 101 to 1000. The input for which the model should be predicted is 1 to 100. The following sections describe in detail the approach to the problem.

Input layer: Two types of input to the network was considered.

A. **Integer input**: This makes the dimension of the input layer as one.
B. **Binary input**: Considering the input data, we would need 10 bits to encode it into binary format. This makes the input layer size as 10.

The **binary format** was chosen for this project. (Integer input was tried, but did not give satisfactory results).

Output layer: The output layer has 4 nodes and the activation function used is Softmax. The loss function used is Categorical Cross Entropy. For getting the class labels, an agrmax() function will be used on the softmax output and a given set of conditions of this class label is evaluated and the corresponding output is printed onto the specified output file, 'Software2.txt'.

Hidden layers: In this assignment, the parameters in the hidden layers have been changed and various trials have been carried out.

Keras has been used to construct the neural network. A dropout regularizer with a probability of 0.1 has been added to each hidden layer to prevent overfitting. The activation function for each layer is selected as Relu.

4. Results and Observations.

The results mentioned here are the following:

- Training Accuracy: Output from Keras is used and the corresponding value of the last epoch was taken as the metric.
- Testing Accuracy: This was measured by comparing the output with the expected ground truths. The correct count was taken and then a ratio with the total was measured as the testing accuracy.
- FizzBuzz Accuracy
- Fizz Accuracy
- Buzz Accuracy

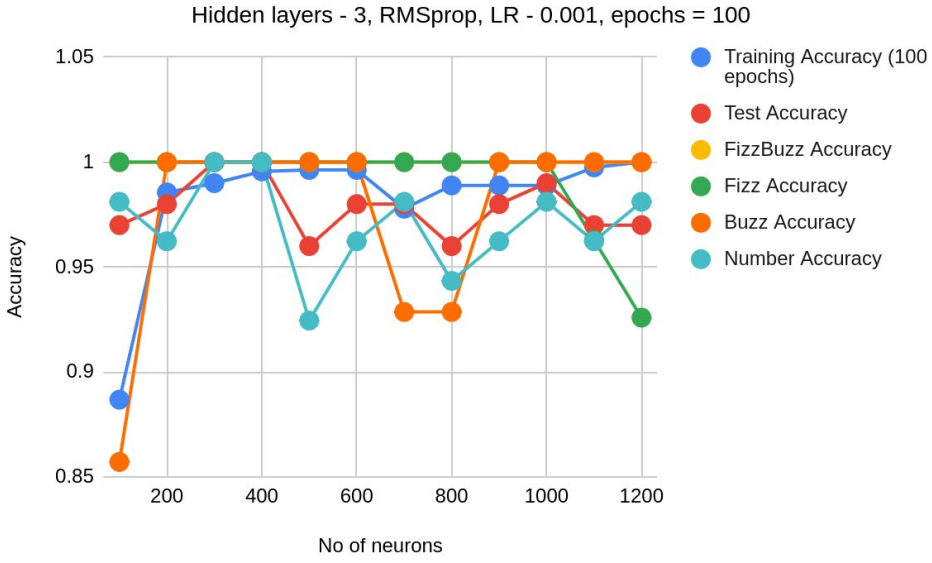
The three metrics above have been generated by counting only the matching output with the ground truth values.

- Number Accuracy: This metric corresponds to where the output is the number itself.

Several trials were tried out, parameters were perturbed and the results have been listed below.

A. Trial 1:

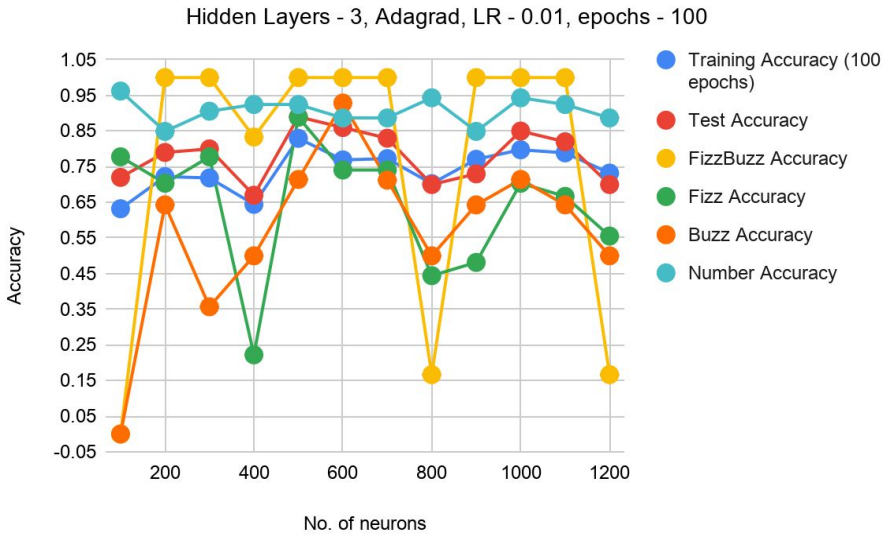
Parameter perturbed: No. of neurons
No. of hidden layers: 3
Optimizer: RMSprop
Loss function: Categorical cross entropy
Learning Rate: 0.001
Number of neurons: 100 to 1200



In the figure above, the training accuracy is increasing as the number of neurons increases. The testing accuracy is comparatively constant over the iterations. The FizzBuzz, Fizz and Buzz accuracies are fairly constant and correct most of the time. The inaccuracies occur in the case of the number accuracies.

B. Trial 2:

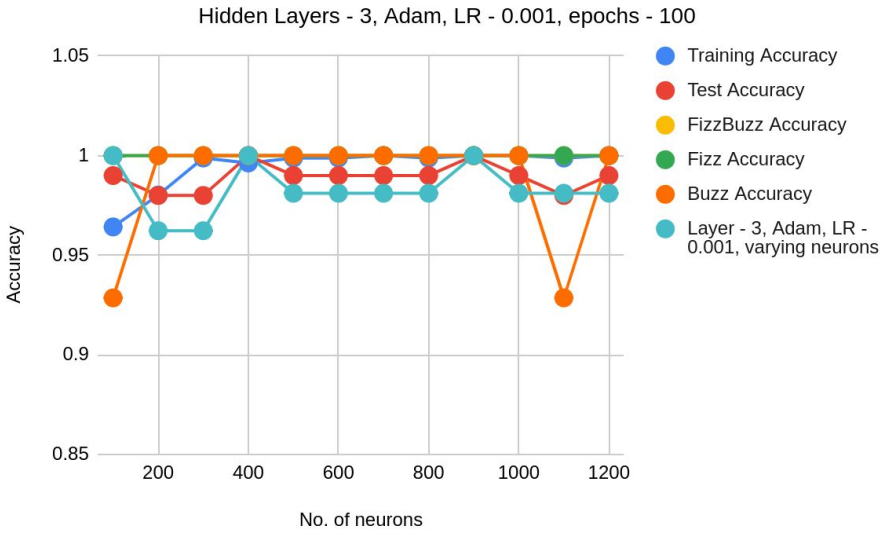
Parameter perturbed: No. of neurons
No. of hidden layers: 3
Optimizer: Adagrad
Loss function: Categorical cross entropy
Learning Rate: 0.01
Number of neurons: 100 to 1200



We can see that both the training and the testing accuracy did not reach 100%. Other accuracy metrics seem to oscillate except number accuracy which is fairly constant throughout. FizzBuzz accuracy hits the 100% mark most of the times, but dips considerably otherwise.

C. Trial 3:

Parameter perturbed: No. of neurons
No. of hidden layers: 3
Optimizer: Adam
Loss function: Categorical cross entropy
Learning Rate: 0.01
Number of neurons: 100 to 1200



Adam gave the best results out of all. The results were very consistent.

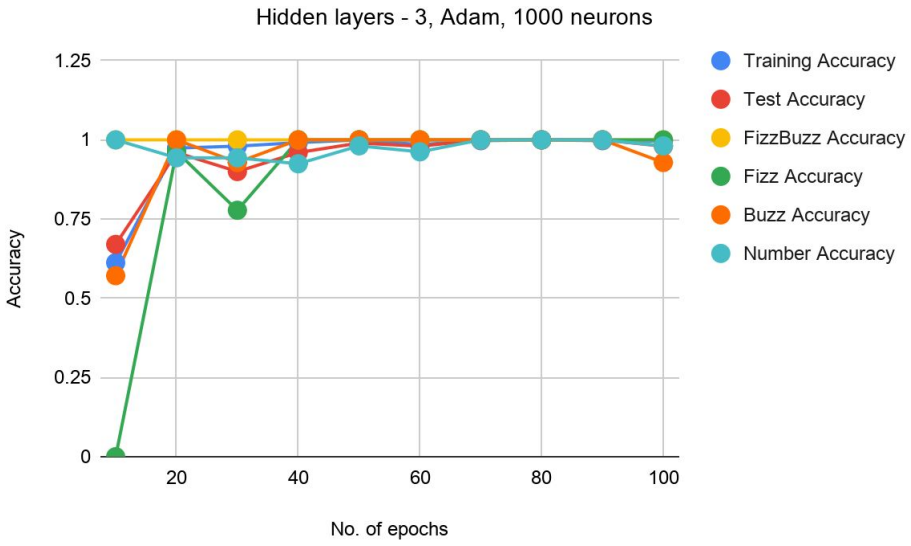
D. Trial 4:

Like the previous trials, the optimizer Stochastic Gradient Descent was also tried. The parameter 'momentum' and 'Nesterov' were kept to 0.0 and False respectively.
The following results were seen for all number of neurons uniformly,
Training accuracy: 53.33%
Testing accuracy: 53%
FizzBuzz accuracy: 0
Fizz accuracy: 0
Buzz accuracy: 0
Number accuracy: 100%

Thus we can see that only the inputs where the outputs were the numbers itself was correctly predicted by the network and the rest were all incorrect.

E. Trial 5:

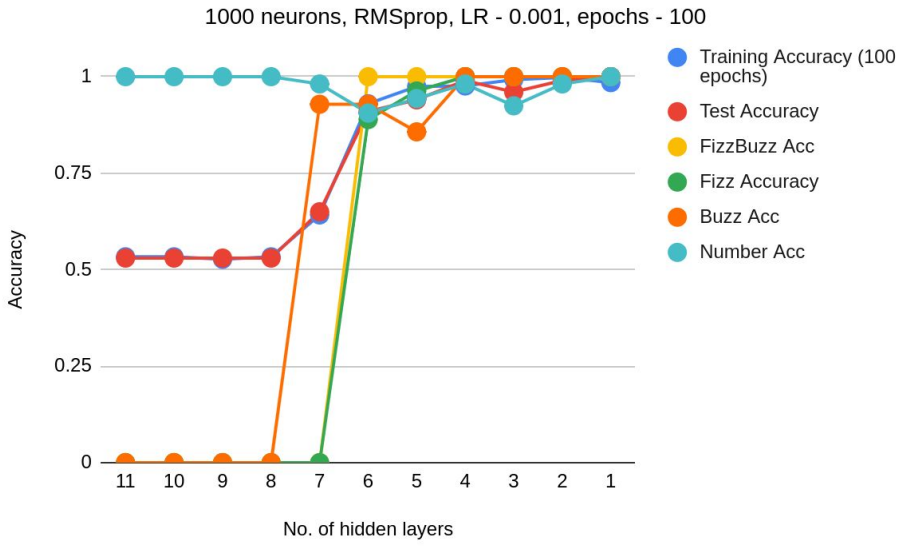
Parameter perturbed: Epochs (10 to 100)
No. of Hidden layers: 3
No. of neurons: 1000
Optimizer - Adam
Loss function: Categorical cross entropy
Learning Rate: 0.01



This trial involved varying the epochs in training. After 20 epochs, the results were fairly constant. This shows that network parameters like number of neurons, optimizer and the learning rates have a much higher impact on the results than the epoch count.

F. Trial 6:

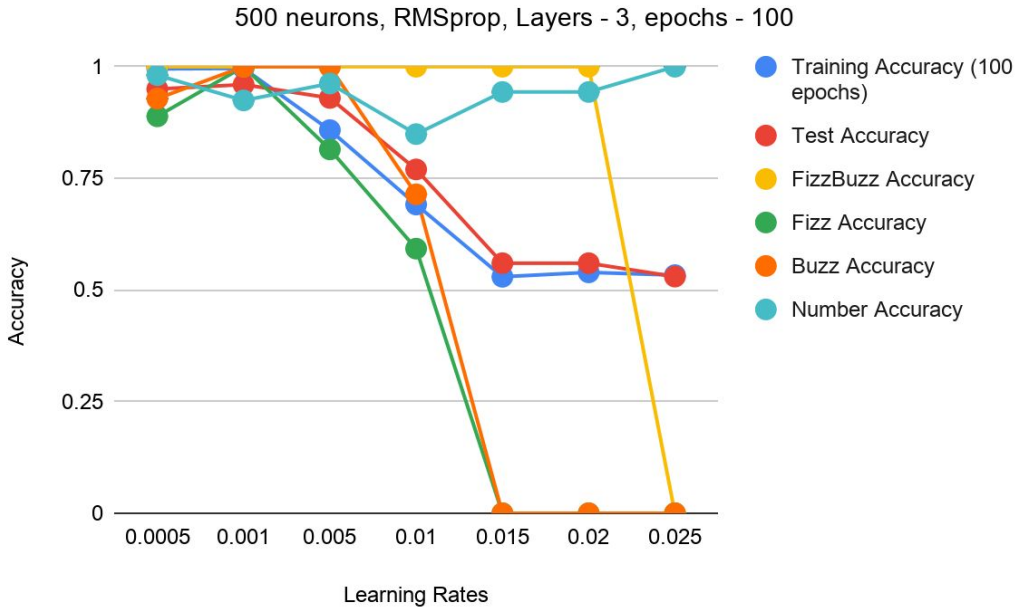
Perturbed parameter: Number of hidden layers
No. of neurons: 1000
Optimizer - RMSprop
Learning Rate: 0.001
Loss function: Categorical cross entropy
Epochs: 100



We can see that after the number of hidden layers increases beyond 5, the accuracy considerably decreases. Thus increasing the number of hidden layers, can decrease the performance of the network.

G. Trial 7:

Perturbed parameter: Learning Rate
No. of neurons: 500
Optimizer - RMSprop
Loss function: Categorical cross entropy
Epochs: 100



Increasing the learning rate substantially decreases the accuracy of the network. But also, some optimizers were seen to perform better with higher learning rates. Adagrad and Adam had a default learning rate of 0.01, while RMSprop had 0.001.

H. Models which gave 100% accuracy for the test input.

Number of hidden layers	Number of neurons per hidden layer	Optimizer	Learning rates	Epochs
3	100	Adam	0.001	100
3	400	Adam	0.001	100
3	900	Adam	0.001	100
3	1000	Adam	0.001	100
3	1000	Adam	0.001	70
3	1000	Adam	0.001	90
3	600	RMSprop	0.001	100
3	500	RMSprop	0.001	100
3	300	RMSprop	0.001	100
3	400	RMSprop	0.001	100

All these models have been saved and saved in the ‘Best_models’ directory.
The model that will be used in main.py is the one marked in green. The following figures represent the plots of the loss and accuracy during the training phase.