FizzBuzz Implementation using Machine Learning

Gagan Suneja
School of Management
University at Buffalo
Buffalo, NY 14260
gagansun@buffalo.edu

Abstract

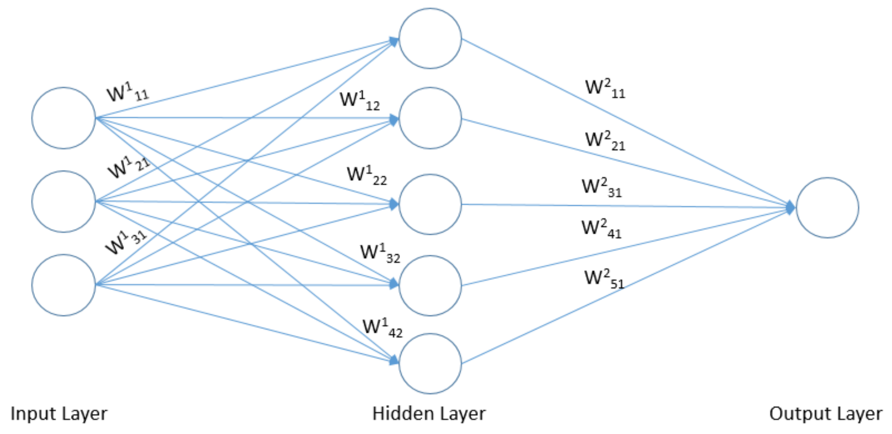
The report discusses about implementation of the FizzBuzz problem using keras (a machine learning library). Further, it captures the information on the performance of a machine learning algorithm, implemented using Sequential Model. Many hyper-parameters like dropout rate, activation function, number of layers, have been fine tuned to get the best accuracy of the program.

1 Introduction

Machine learning involves inculcating decision making capabilities to computers or software programs so that they can learn further and decide on their own. It is mainly of three types- *Supervised Learning*, *Unsupervised Learning* and *Reinforcement Learning*. The task of FizzBuzz has been implemented using Supervised Learning, so the scope of the report will be limited to Supervised Learning only. Supervised learning is a branch of machine learning where a machine learning algorithm learns from solved problem set (also called training data) to develop decision making skills. It is then tested with the test data to check for correct output.

Artificial Neural Network

Artificial Neural Networks(ANNs) are an important class of machine learning algorithms that exhibit behavior analogous to a human brain. A simple neural network consists of 3 components – *Input Layer*, *Hidden Layer* and *Output Layer.* There can be multiple hidden layers in the network and each hidden layer comprises of nodes which are called neurons. A neuron takes input from previous layer, processes it with a weight W and an activation function A and passes the output to next layer.



2 Fizz Buzz Implementation

The machine learning problem has been implemented using Sequential model in Keras, an open source neural network library in python. At first the training dataset is created with input integer from 101 to 1000 using the logic based approach and is stored in training.csv file.

Then, a Sequential Model is created in which for the first hidden layer, a dense Layer with 256 nodes is added and an activation function 'relu' is applied to it. A dropout layer of 20% is added to the model. Now, for the output layer, a dense layer of 4 nodes (as we only have 4 classes - Fizz, Buzz, FizzBuzz, and Other) is added along with an activation function 'softmax'. Here softmax function is used for output layer, in order to calculate the probabilities for each of the four output classes. It squashes the output of each node to be between 0 and 1 and also divides each output such that the sum of all outputs is 1. In the end, the model is compiled using compile function in keras library using 'rmsprop' optimizer with categorical cross entropy. Here categorical cross entropy is used as the output has 4 types of classes.

Further, while running the model, the training data is read and is processed first to convert the integer representation of input numbers into binary representation. After the model has run, various Training and validation graphs are created to measure how accurate the model is.

3 Observations and Graphs

Below are the observations recorded-

Relationship between number of first dense layer nodes and Accuracy

With dropout=0.2, activation function used = softmax, and rmsprop as the optimizer, below are the observations for the number of First Dense Layer Nodes and the Accuracy-

Table 1: Observations recorded for First Dense Layer Nodes and Accuracy

No. of first dense layer nodes	Accuracy
128	72
256	88
512	96
1024	98
1500	97
2048	98

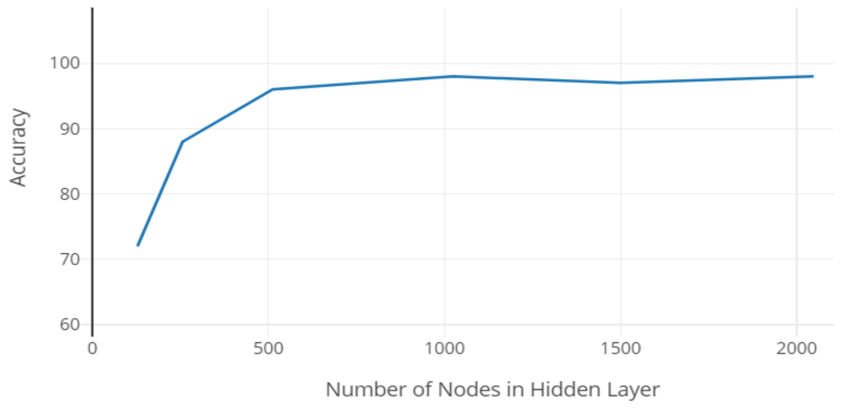


Fig.1: Accuracy V/S # Nodes in Hidden Layer

The maximum Accuracy (98) was obtained at 1024 nodes. Now on keeping the First Dense Layer Nodes constant at 1024, below were the observations recorded for Dropout rate and Accuracy.

Table 2: Observations recorded for Dropout rate and Accuracy with # first dense layer nodes=1024 and

Dropout Rate(%)	Accuracy
10	98
20	98
30	94
40	89
50	92
60	88
70	79
80	68
90	60

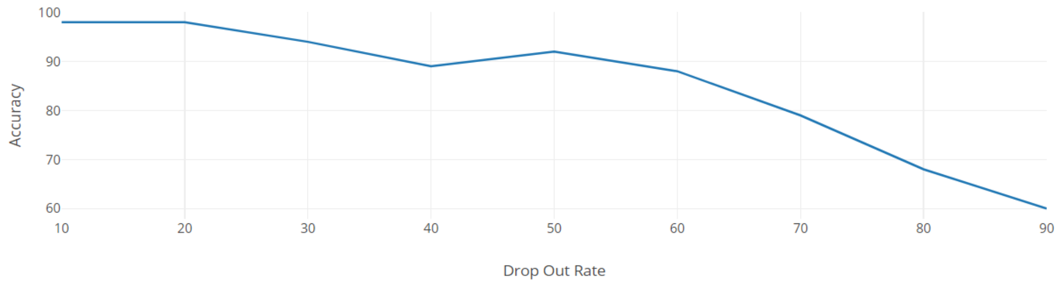


Fig.2: Accuracy vs Dropout Rate

Table 3: Observations recorded with Dropout rate at 20%, hidden layer nodes at 1024

Validation Data Split	Accuracy
0.1	99
0.2	98

0.3	95
0.4	89
0.5	85
0.6	55
0.7	53
0.8	53
0.9	47

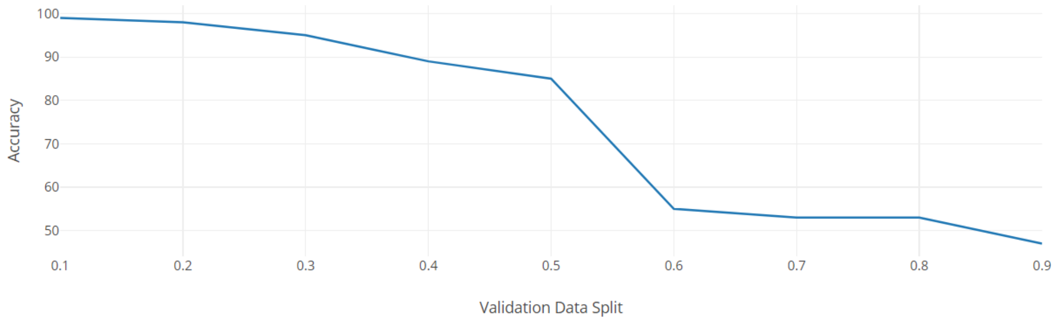


Fig.3 Accuracy vs. Validation Data Split

Also, on changing the Activation function to sigmoid, with dropout=0.2, similar trend was observed in Accuracy with maximum Accuracy observed at 98 with 1024 nodes in the first dense layer. Further on changing the optimizer from ‘rmsprop’ to ‘sgd’, maximum accuracy of 99 was observed with 1024 nodes in the first dense layer.

4 Inference and Conclusion

For dropout rate at 20%, activation function used as softmax, and using rmsprop as the optimizer, the maximum accuracy is achieved at 1024 nodes and is 99 for validation split 0.1. From Table 1 and Fig1, it can be stated that as the number of nodes increase in the hidden layer, the accuracy also increases.

From Table 2 and Fig2, it can be stated that dropout rate and accuracy are inversely proportional to each other. As the dropout rate increases, the accuracy also decreases.

Also, from Table 3 and Fig 3, it can be stated that as the validation data split increases, the accuracy decreases. The maximum accuracy is achieved with validation data split at 0.1.

Also, maximum accuracy (99) can be observed with optimizer set to ‘sgd’ with dropout rate of 20%.