# Introduction to Machine Learning
# Project 1.1
# Report

**UBID: sppandey@buffalo.edu**
**PERSON NO: 50288608**

**Objective**: To compare the logic based approach (Software 1.0 - Python) and the Machine Learning approach (Software 2.0 - Python + Tensorflow) to solve the FizzBuzz Problem.

**FizzBuzz Overview**: For any given integer in a range, if the integer is divisible by 3, it should print Fizz and if it divisible by 5 it should print Buzz, additionally if a number is divisible by both 3 and 5 then it should print FizzBuzz. On the contrary, if the number is neither divisible by 3 nor 5 it should simply print the supplied integer.

**Software 1.0:** The python code runs a basic if-else check and prints out the result. However, this software version plays an important role in creating the training and testing dataset for the model training in Software 2.0.

**Software 2.0:** In this version of FizzBuzz, the performance and accuracy of the output obtained from the Tensorflow code has been highlighted.

The Report consists of statistics and different accuracies obtained by changing the hyper-parameters such as epochs, learning rate, optimizers and activation functions for the different scenarios. At the end of the report I have compared performance of different Optimizers given the same input and activation Function.

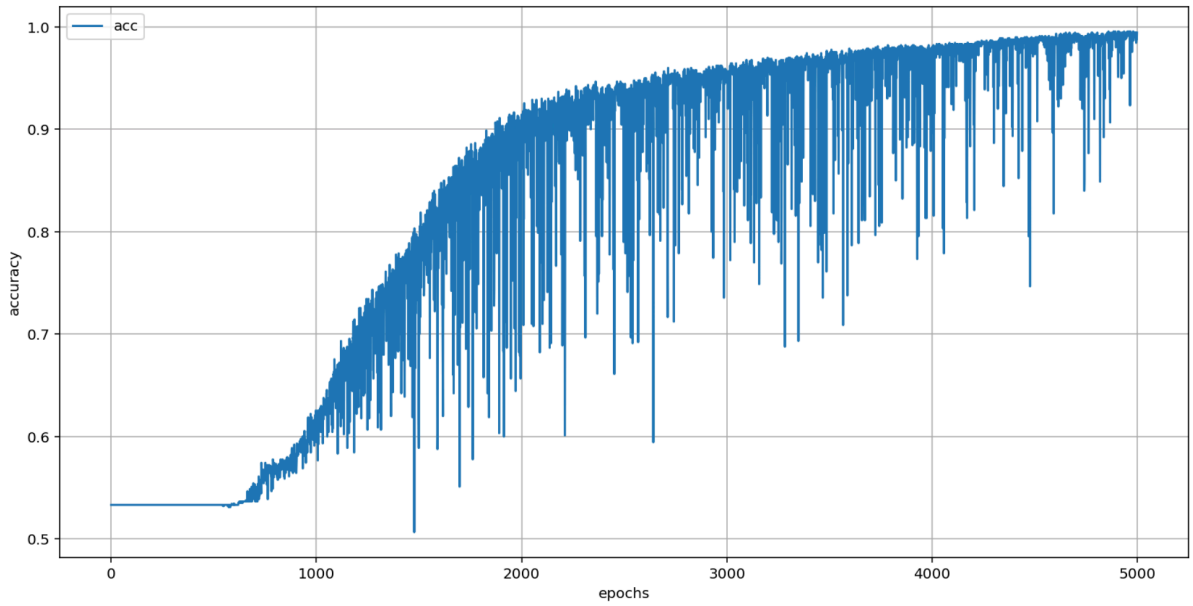
## Scenario 1:

Activation Function(Input): ReLU
Optimizer: GradientDescent

**Accuracy over different learning rates:**

| Epoch | Batch Size | Learning Rate | Accuracy |
|---|---|---|---|
| 5000 | 128 | 0.05 | 92 |
| 5000 | 128 | 0.0005 | 53 |
| 5000 | 128 | 0.5 | 91 |

**Graph for best Accuracy:** Learning Rate = 0.05




**Observation:** For a fixed number of epochs and batch size, the accuracy was the highest for a learning rate of 0.05. The accuracy at learning rate 0.5 gave results close to that of 0.05 rate. Accuracy was highly affected when learning rate was dropped to as low as 0. 0005.Thus a decrease in learning rate with constant number of epochs impacted the accuracy of the above model.

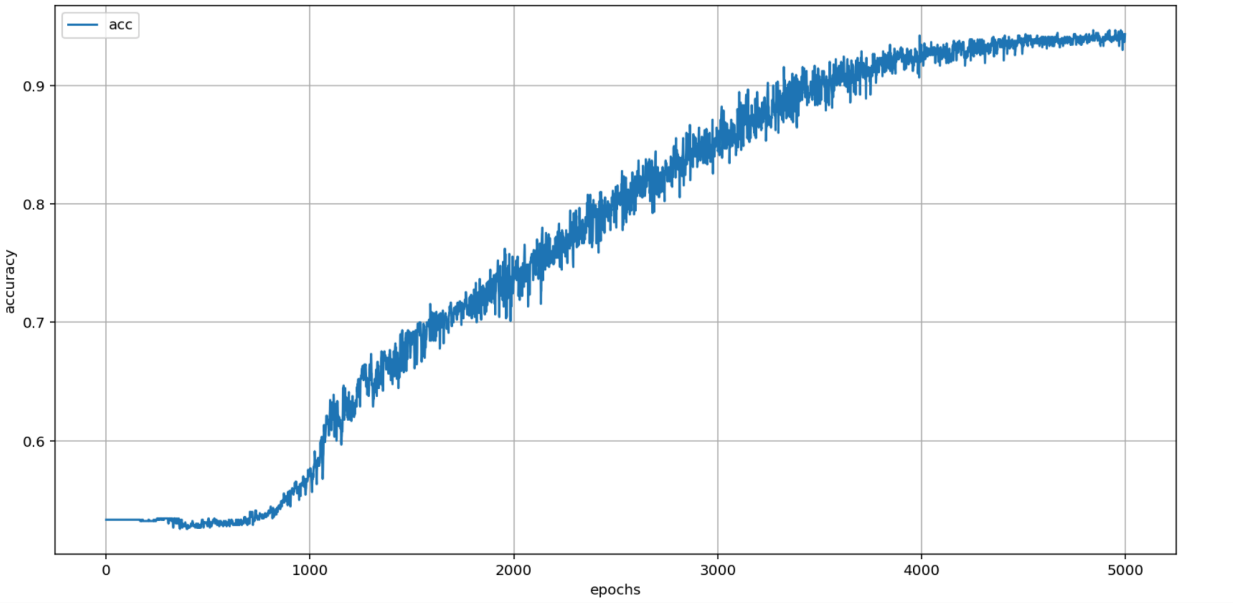
## Scenario 2:

Activation Function(Input): ReLu
Optimizer: Adam

**Accuracy over different learning rates:**

| Epoch | Batch Size | Learning Rate | Accuracy |
|---|---|---|---|
| 5000 | 128 | 0.05 | 52 |
| 5000 | 128 | 0.0005 | 89 |
| 5000 | 128 | 0.5 | 47 |

**Graph for best Accuracy:** Learning Rate = 0.0005




**Observation:** For a constant number of epochs and batch size, the accuracy was the highest for a learning rate of 0.0005. Gradual increase in learning rate showed a fall in the accuracy when AdamOptimizer was used with ReLU activation function

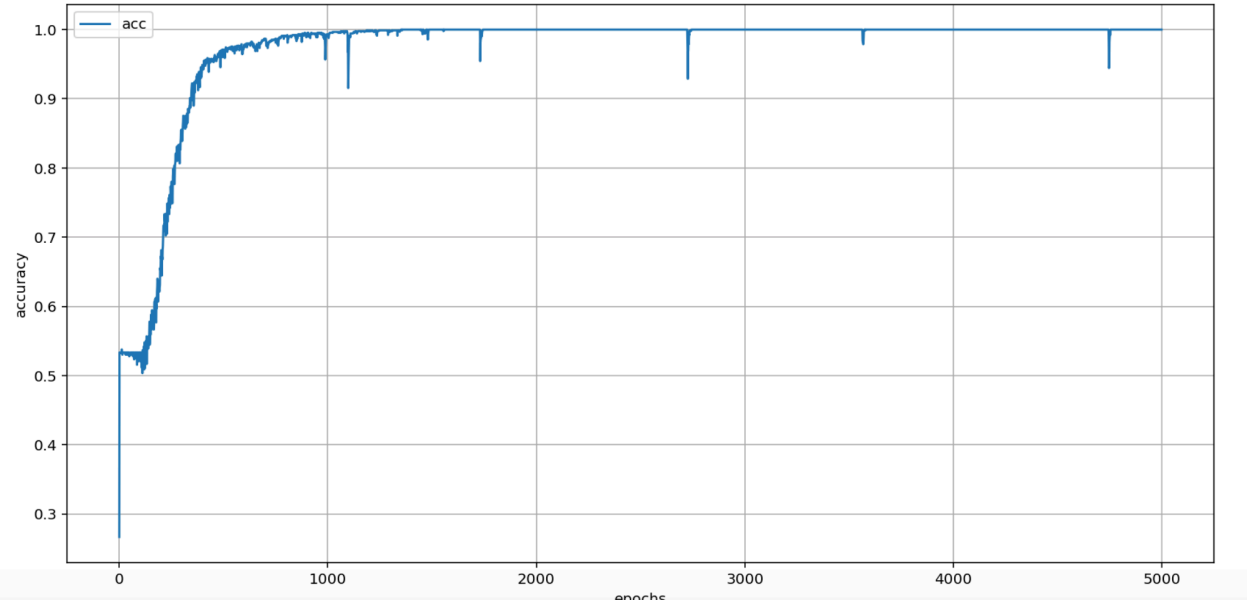
## Scenario 3:

Activation Function(Input): Sigmoid
Optimizer: Adam

### Accuracy over different learning rates:

| Epoch | Batch Size | Learning Rate | Accuracy |
|---|---|---|---|
| 5000 | 128 | 0.05 | 88 |
| 5000 | 128 | 0.0005 | 53 |
| 5000 | 128 | 0.5 | 51 |

**Graph for best Accuracy:** Learning Rate = 0.05



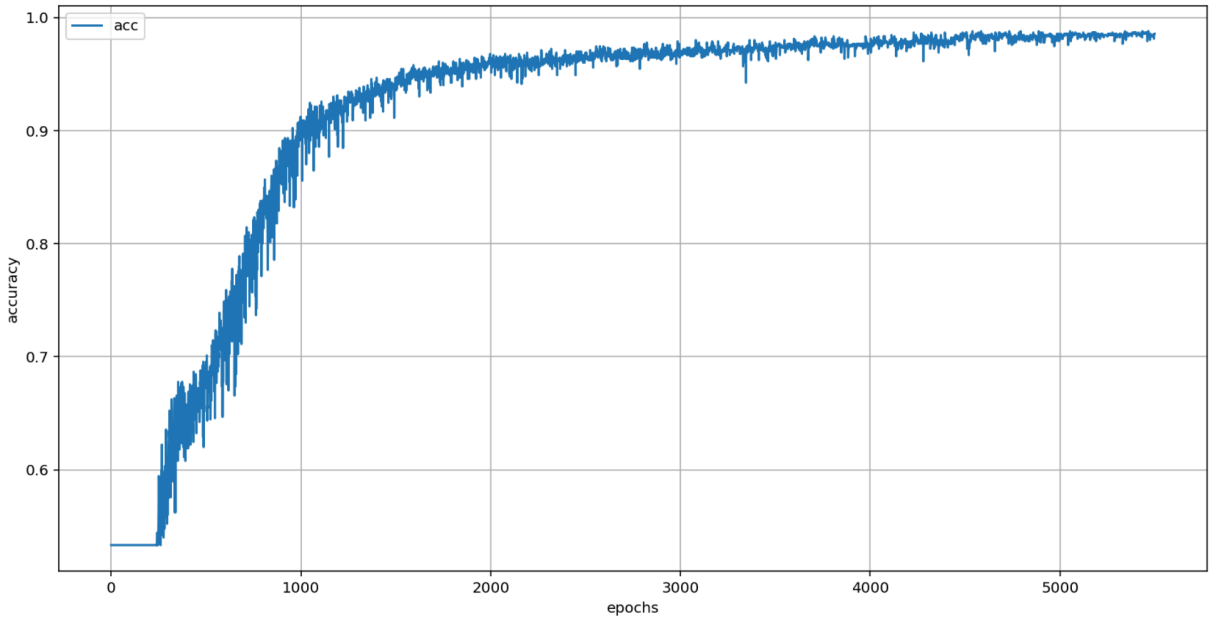

**Observation:** For a constant number of epochs and batch size, the accuracy was the highest for a learning rate of 0.05. Repeated training and testing using the sigmoid-Adam pair gave stable accuracy of about close to 50.

## Comparison of Optimizers using the ReLU activation Function

### Accuracy observed:

| Model (Activation Function–Optimizer) | Epoch | BatchSize | Learning Rate | Accuracy |
|---|---|---|---|---|
| ReLU-Adagrad | 5500 | 128 | 0.05 | 91 |
| ReLU-RmsProp | 5500 | 128 | 0.05 | 66 |
| ReLU-Adadelta | 5500 | 128 | 0.05 | 53 |

### Graph for best Accuracy: Adagrad Optimizer outperformed other two optimizers



Implemented Code:  For a given number of Epochs, batch size and learning rate Adagrad Optimizer and ReLU activation function had the accuracy.

**References:**

[1] https://www.tensorflow.org/api_docs/python/

[2]  https://en.wikipedia.org/wiki/Multilayer_perceptron

[3] http://ruder.io/optimizing-gradient-descent/index.html#adagrad

[4] https://matplotlib.org/api/