Project 1.1: Software 1.0 Versus Software 2.0

KRISHNA SEHGAL
UB ID: 50291124

1. Objective

The project is to compare two problem-solving approaches to software development:

- The logic-based approach (Software 1.0)
- The machine learning approach (Software 2.0).

2. Task

We consider the task of FizzBuzz. 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 an integer is not divisible by 3 or 5 or 15, it simply prints the input as output.

2.1 Software 1.0

We are making use of for loop to run across numbers from 1 to 100 and if-else statement i.e if number is divisible by 3 print 'fizz' if number is divisible by 5 print 'buzz' if number is divisible by 15 print 'fizzbuzz' else print 'other'

Code for Software 1.0

```
for number in range(1, 100):
    if number % 3 == 0 and num % 5 == 0:
        print('FizzBuzz')
    elif number % 3 == 0:
        print('Fizz')
    elif number % 5 == 0:
        print('Buzz')
    else:
        print(number)
```

2.2 Software 2.0

We are solving Fizz-buzz problem by making use of Classification method, which is a type of Supervised Machine Learning model that uses labeled data for learning and makes prediction to classify a new data input into a labeled group.

First we are creating two CSV files, one for training and the other for testing. The training CSV file contains labeled data having input as numbers from 101-1000 and output in form of Fizz, Buzz, Fizz-buzz and other and the testing CSV file contains numbers from 1-100 and corresponding outputs.
We will be training our classifier by giving training data as input and once our model is generated, testing data file will be used to verify outcomes of our model. After the training phase we will be checking accuracy of our model by comparing expected results with the output of our model.

After creation of data file we will convert each number into its Binary form since we have a large dataset so it is difficult to manage. Since we have numbers from 101-1000 so 10 bits will be able to cover thousand numbers as 2^{10}=1024. After converting integers into Binary we will make use of right shift operator to encode the dataset making it easy to process.

We have used the concept of neural networks where we have an input layer of ten perceptron's and an output layer of four perceptron's representing our four outputs. In between our input and output layer we have a number of hidden layers.
Now, ten input layer perceptron's are connected to the perceptron's of hidden layers having weights. These weights are given a random value and we perform matrix multiplication of input layer with these hidden layer weights.
The output is passed through Activation function such as sigmoid, relu, tanh defining the output of the node for given set of inputs.
We use error function/Cost function to check how far our output is from the expected value. If the difference of expected and predicted value is large, neurons will learn faster in case of Cross Entropy.
We have used Gradient Descent as optimizer to minimize the cost function by changing the value of hidden layer weights. We can make use of other Optimizers as well such as SGD, Adagrad, Adadelta, Rmsprop and Adam.
The method of Back-propagation is used to reiterate across the layers multiple times so as to reduce error and improve efficiency.
The number of iterations is represented by number of epochs.
Batch size refers to the number of inputs fed to neural network in single iteration. By changing the batch size our accuracy gets changed.
Learning Rate refers to the amount of time taken by our algorithm to learn. If we reduce value of learning rate, classifier takes longer time to learn but with an improved accuracy.
After encoding of data and labels, input values are fed to the classifier and training of classifier takes place.
After training process is complete we perform the testing phase for that we are using numbers from 1-100 as our test cases. When testing data is input to classifier, the output is in encoded form as 0, 1, 2 and 3. We decode into Fizz, Buzz, Fizz-Buzz and other. After that Accuracy is found by comparing expected output and neural network output based on which we are plotting a graph.

3. Software Used

- Anaconda – Navigator
- Jupyter Notebook
- Python 3.6

4. Libraries Used

- Numpy
- Tensor Flow
- Pandas
- Keras
- Matplot
- Tqdm

5. Hyper-parameters Used

- Activation Function

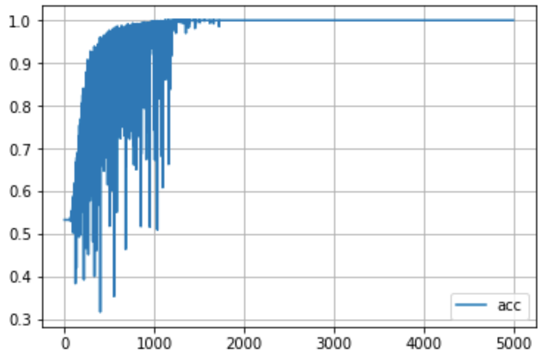
    - Rectified Linear Unit
    - Sigmoid
    - Hyperbolic Tangent

- Optimization Method
  - Simple Gradient Decent Optimizer
  - Adagrad Optimizer
  - Adadelta Optimizer
  - Adam Optimizer
- Learning Rate
- Number of Epochs
- Number of Hidden Neuron Layers
- Batch Size

6. Graphical Representation by varying Hyper-parameters

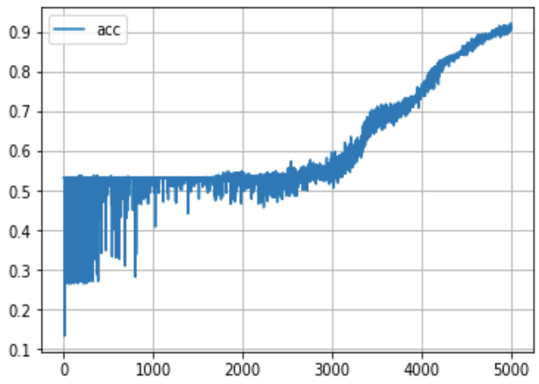
Hyper-parameters = [Activation Function, Optimizer, Learning Rate, Number of Epochs, Number of hidden neuron Layer, Batch Size]

6.1 [Rectified Linear Unit (Relu), Simple Gradient Descent, 0.5, 5000, 100, 125]



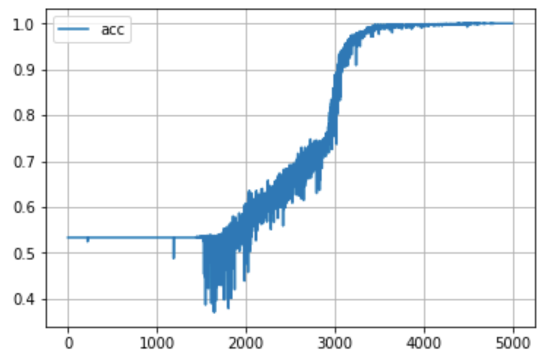
```
Errors: 3  Correct :97
Testing Accuracy: 97.0
```

6.2 [Sigmoid, Simple Gradient Descent, 0.5, 5000, 100, 125]



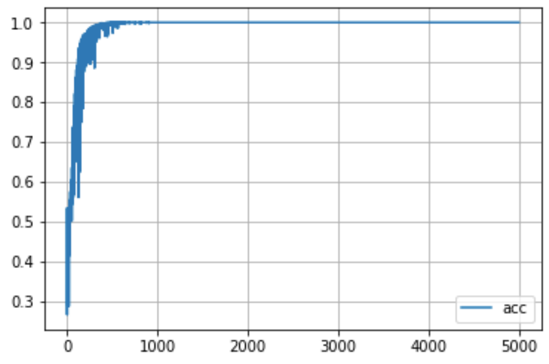
```
Errors: 15  Correct :85
Testing Accuracy: 85.0
```

6.3 [Hyperbolic Tangent (tanh), Simple Gradient Descent, 0.5, 5000, 100, 125]



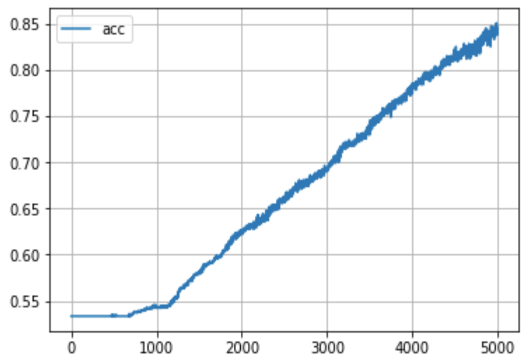
```
Errors: 21  Correct :79
Testing Accuracy: 79.0
```

6.4 [Rectified Linear Unit (Relu), Adagrad Optimizer, 0.5, 5000, 100, 125]



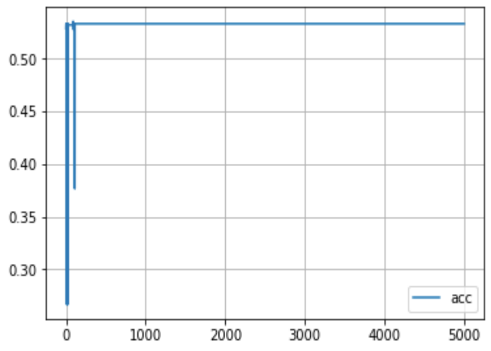
```
Errors: 2  Correct :98
Testing Accuracy: 98.0
```

6.5 [Rectified Linear Unit (Relu), Adadelta Optimizer, 0.5, 5000, 100, 125]



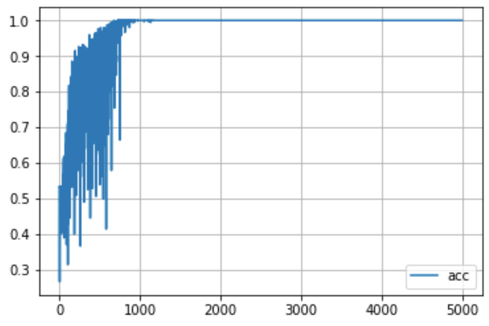
```
Errors: 28  Correct :72
Testing Accuracy: 72.0
```

6.6 [Rectified Linear Unit (Relu), Adam Optimizer, 0.5, 5000, 100, 125]



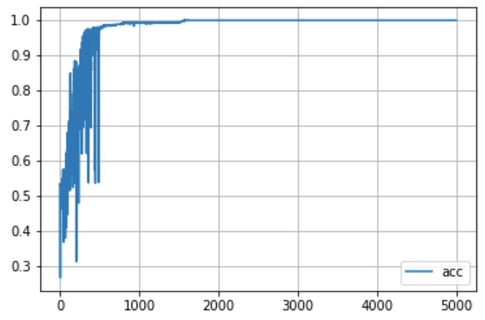
```
Errors: 48  Correct :52
Testing Accuracy: 52.0
```

6.7 [Rectified Linear Unit (Relu), Simple Gradient Descent, 1, 5000, 100, 125]



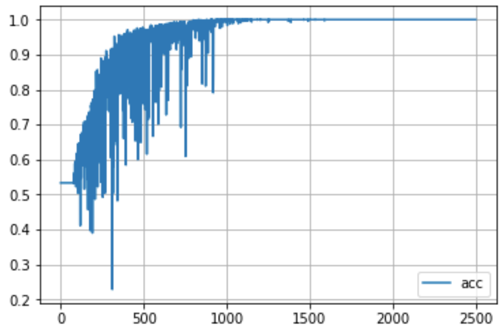
```
Errors: 10  Correct :90
Testing Accuracy: 90.0
```

6.8 [Rectified Linear Unit (Relu), Simple Gradient Descent, 2, 5000, 100, 125]



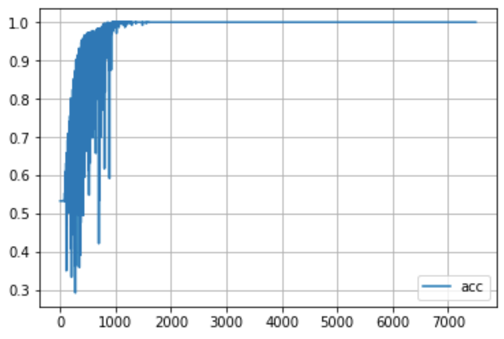
```
Errors: 27  Correct :73
Testing Accuracy: 73.0
```

6.9 [Rectified Linear Unit (Relu), Simple Gradient Descent, 0.5, 2500, 100, 125]



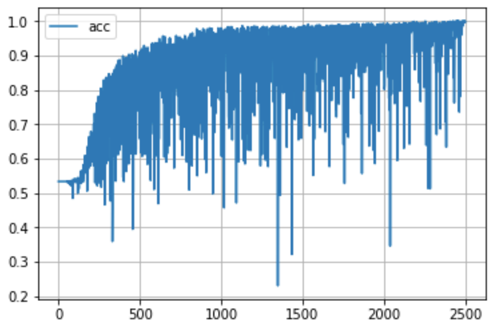
```
Errors: 6  Correct :94
Testing Accuracy: 94.0
```

6.10 [Rectified Linear Unit (Relu), Simple Gradient Descent, 0.5, 7500, 100, 125]



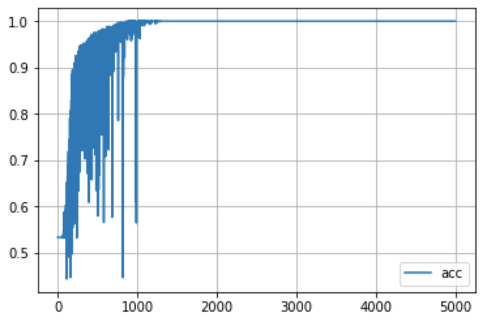
```
Errors: 8  Correct :92
Testing Accuracy: 92.0
```

6.11 [Rectified Linear Unit (Relu), Simple Gradient Descent, 0.5, 5000, 50, 125]



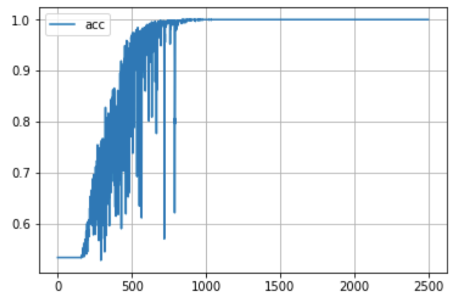
```
Errors: 17  Correct :83
Testing Accuracy: 83.0
```

6.12 [Rectified Linear Unit (Relu), Simple Gradient Descent, 0.5, 5000, 150, 125]



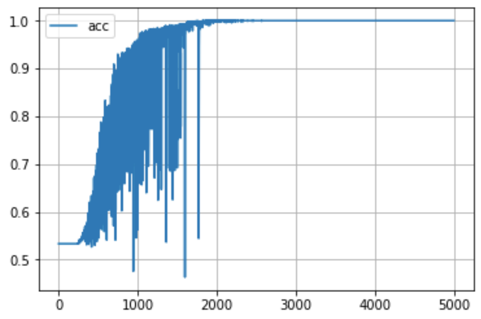
```
Errors: 3  Correct :97
Testing Accuracy: 97.0
```

6.13 [Rectified Linear Unit (Relu), Simple Gradient Descent, 0.5, 5000, 100, 150]



```
Errors: 1  Correct :99
Testing Accuracy: 99.0
```

6.14 [Rectified Linear Unit (Relu), Simple Gradient Descent, 0.5, 5000, 100, 300]



```
Errors: 8  Correct :92
Testing Accuracy: 92.0
```

7. Tabular representation of result with varying Hyper-parameters

Activation Function	Optimizer	Learning Rate	Number of Epochs	Batch Size	Number of Hidden Layers	Accuracy
Relu	Gradient Descent	0.5	5000	100	125	97
Sigmoid	Gradient Descent	0.5	5000	100	125	85
Tanh	Gradient Descent	0.5	5000	100	125	79
Relu	Adagrad	0.5	5000	100	125	98
Relu	Adadelta	0.5	5000	100	125	72

Relu	Adam	0.5	5000	100	125	52
Relu	Gradient Descent	1	5000	100	125	90
Relu	Gradient Descent	2	5000	100	125	73
Relu	Gradient Descent	0.5	2500	100	125	94
Relu	Gradient Descent	0.5	7500	100	125	92
Relu	Gradient Descent	0.5	5000	50	125	83
Relu	Gradient Descent	0.5	5000	150	125	97
Relu	Gradient Descent	0.5	5000	100	125	99
Relu	Gradient Descent	0.5	5000	100	125	92

8. Result

We have compared two problem-solving approaches to software development the logic based and the machine learning approach. In the machine learning approach we have used tensor flow library of python and on changing hyper-parameters we have found that the accuracy of our model comes out to be maximum when we have following values of the hyper-parameters:

- Activation Function: Rectified Linear Unit
- Optimizer: Simple Gradient Descent
- Learning rate: 0.5
- Number of Epochs: 5000
- Number of hidden neuron layer: 100
- Batch Size: 125

Our Accuracy comes out to be 99%.

9. Conclusion

We have used the logic-based approach and machine learning approach to solve the problem of Fizz-buzz.