CSE 474/574: Introduction to Machine Learning

University at Buffalo

Keming Kuang (kemingku)

personal #: 50161776

September 17, 2018

Contents

1 Description

1.1 Objective

In this project, we will compare two different approaches to solve FizzBuzz: Software 1.0 and Software 2.0. We will also get familiar with python and machine learning framework.

1.2 Task

In the FizzBuzz, an integer divisible by 3 is printed as "Fizz", and an integer divisible by 5 is printed as "Buzz". An integer divisible by both 3 and 5 is printed as "FizzBuzz". In any other case, The output will print "Other".

1.3 Software 1.0

We will use simple if-then-else statement using modulo arithmetic to implement the logic in Python. Input value will vary from 0 to 100.

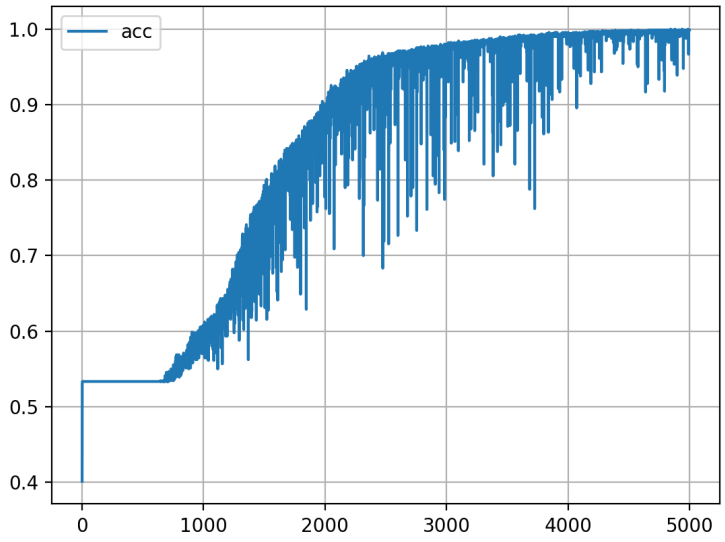
1.4 Software 2.0

Different from Software 1.0, we will use machine learning to solve FizzBuzz. By avoiding testing set 0-100, we will create a training data set for number ranging from 101-1000. By using the training set, we will use tensor-flow to predict 0-100.

2 Performance

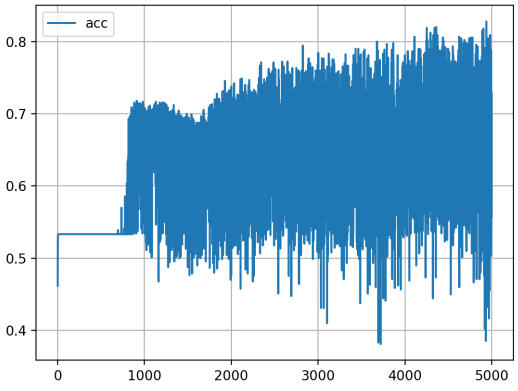
In this section, I will study the Performance when the hyper parameter and activation functions change.
By default, I will use
number of hidden neurons layer 1 = 100
Learning rate = 0.05
number of epochs = 5000
batch size = 128

Testing accuracy = 91%

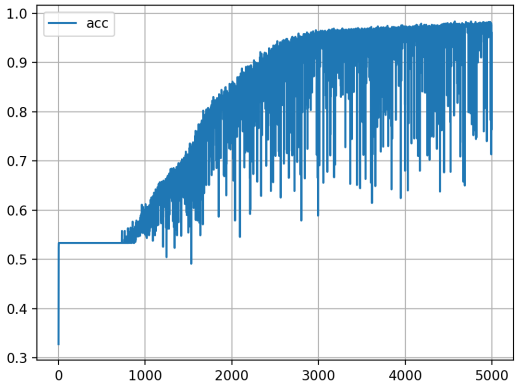


2.1 Number of Hidden Neurons Layer

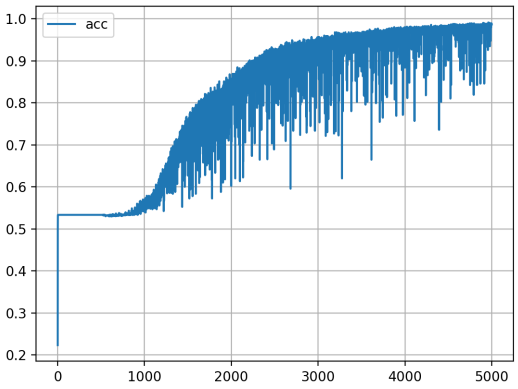
Number of hidden neurons layer = 10
Testing accuracy = 48%



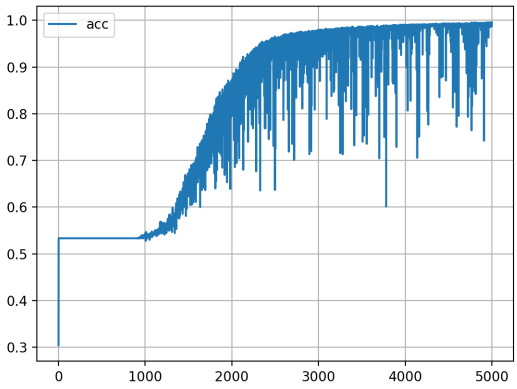
Number of hidden neurons layer = 50
Testing accuracy = 87%



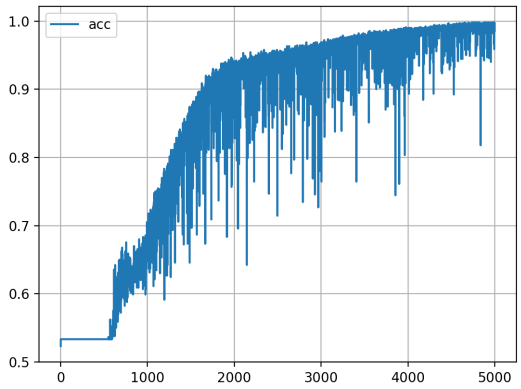
Number of hidden neurons layer = 80
Testing accuracy = 88%



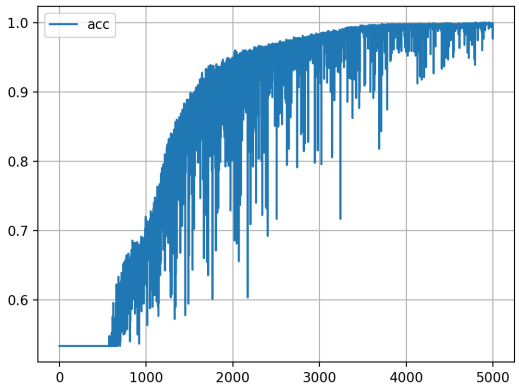
Number of hidden neurons layer = 100
Testing accuracy = 90%



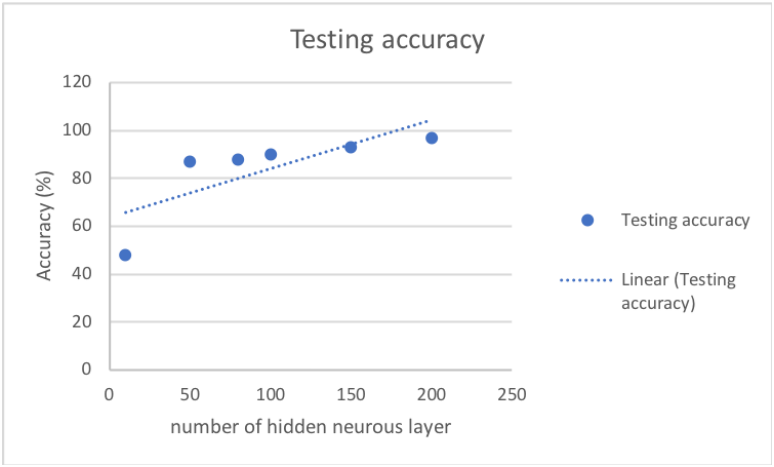
Number of hidden neurons layer = 150
Testing accuracy = 93%



Number of hidden neurons layer = 200
Testing accuracy = 97%



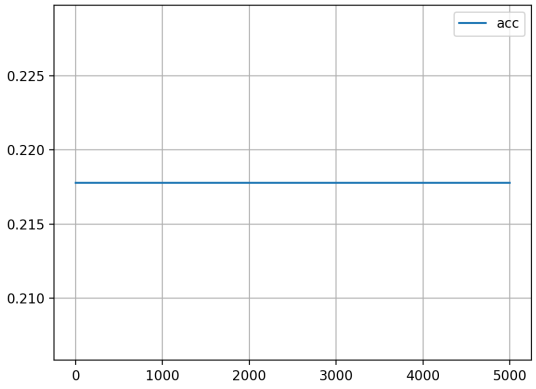
2.1.1 Conclusion



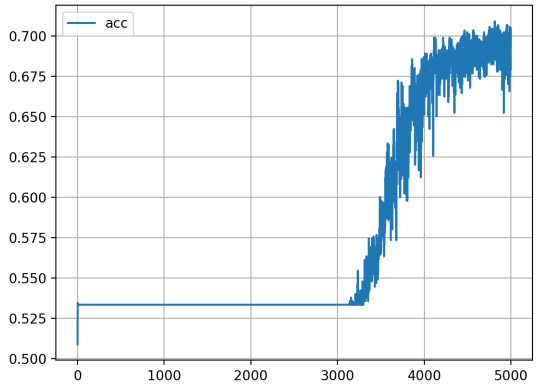
From the graphic shown above, we can see the Testing accuracy will gradually increase when the number of hidden neurons layer increase.

2.2 Leaning Rate

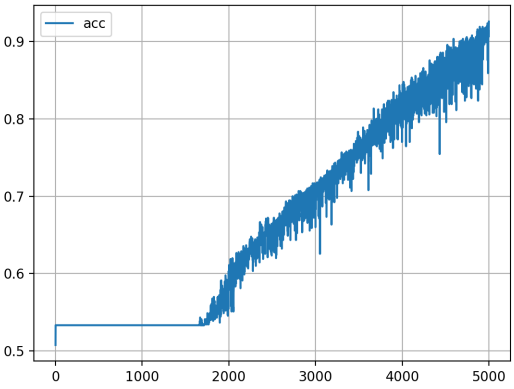
Learning rate = 0
Testing accuracy = 10%



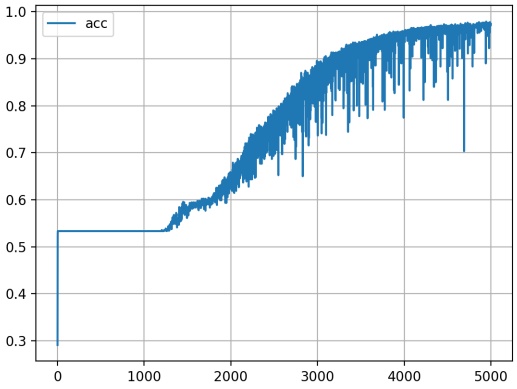
Learning rate = 0.01
Testing accuracy = 69%



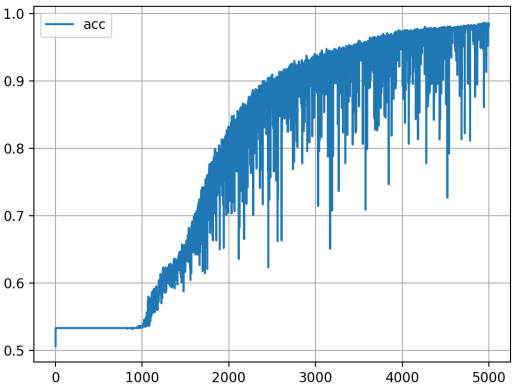
Learning rate = 0.02
Testing accuracy = 90%



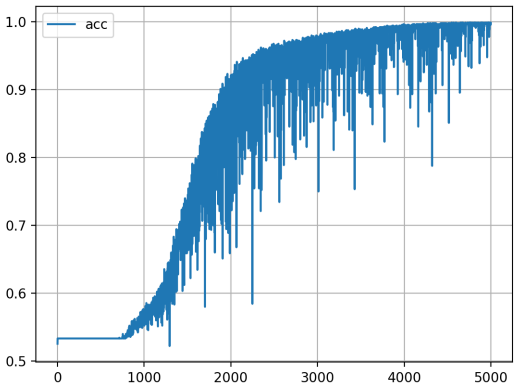
Learning rate = 0.03
Testing accuracy =97 %



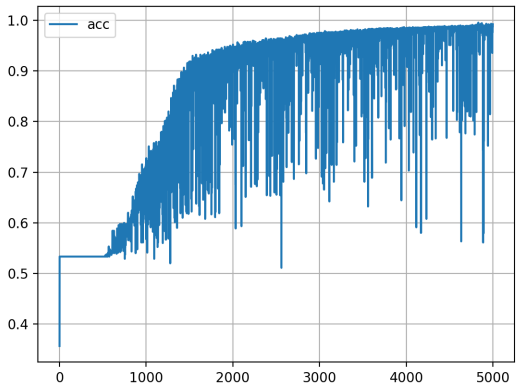
Learning rate = 0.04
Testing accuracy = 93%



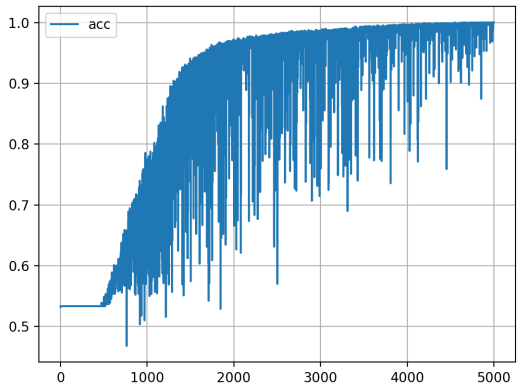
Learning rate = 0.05
Testing accuracy =97 %



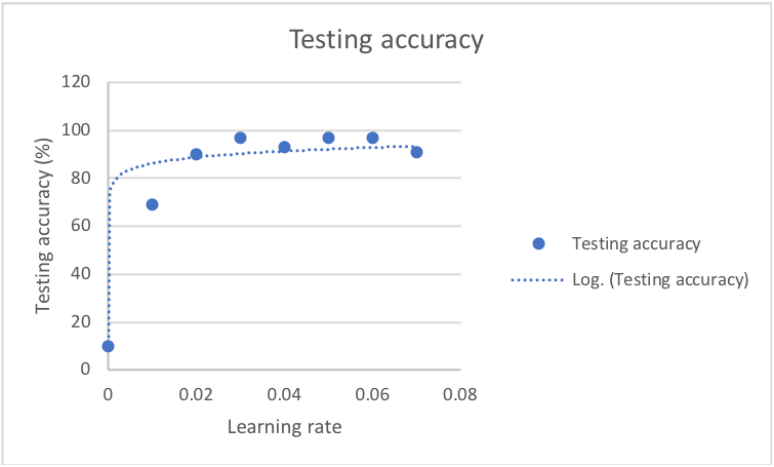
Learning rate = 0.06
Testing accuracy = 97%



Learning rate = 0.07
Testing accuracy =91%



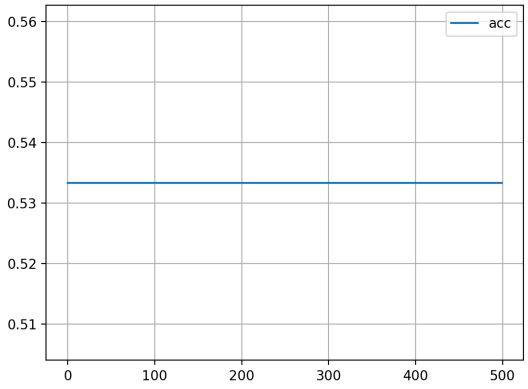
2.2.1 Conclusion



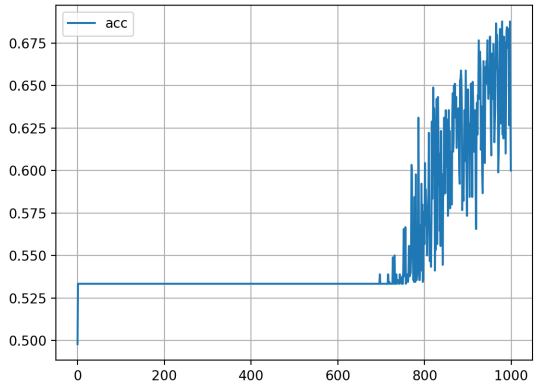
From the graphic shown above, we can see the increase rate of testing accuracy rapidly changes during the increasing of Learning rate. We know that learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradients. From the graphic we know that when the learning rate is low, the testing accuracy will stay low. When the learning rate is high, an increasing in learning rate will only slightly increase test accuracy.

2.3 Number of Epochs

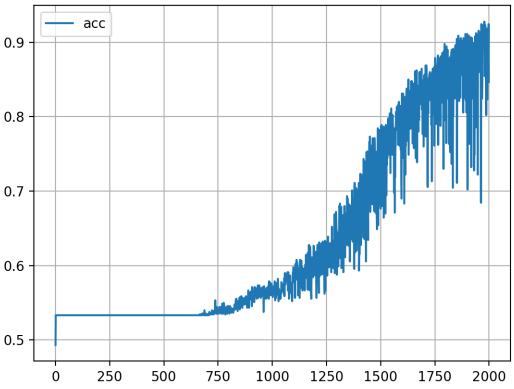
Number of epochs = 500
Testing accuracy = 53%



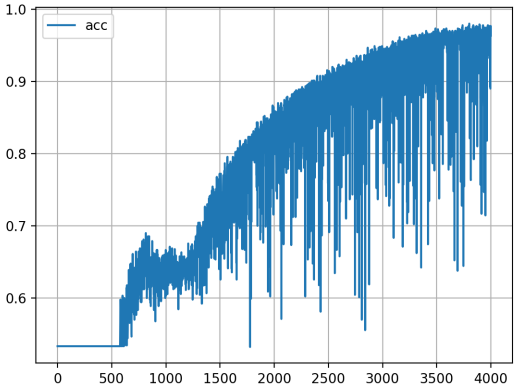
Number of epochs = 1000
Testing accuracy = 58%



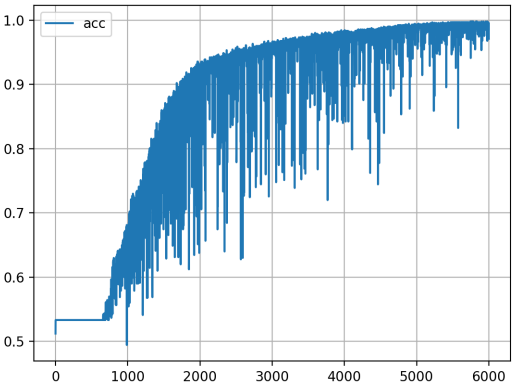
Number of epochs = 2000
Testing accuracy = 84%



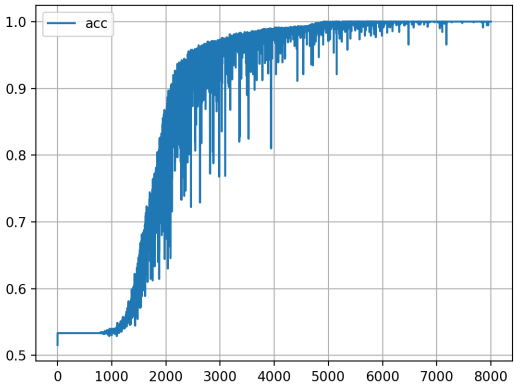
Number of epochs = 4000
Testing accuracy =90 %



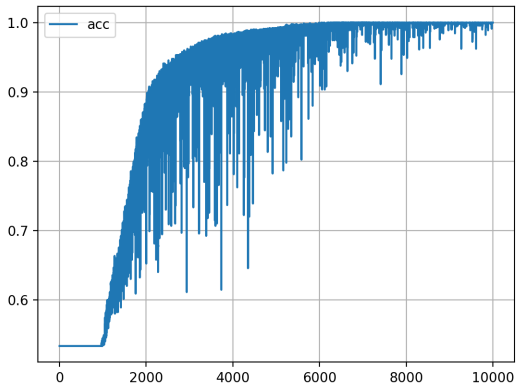
Number of epochs = 6000
Testing accuracy = 92%



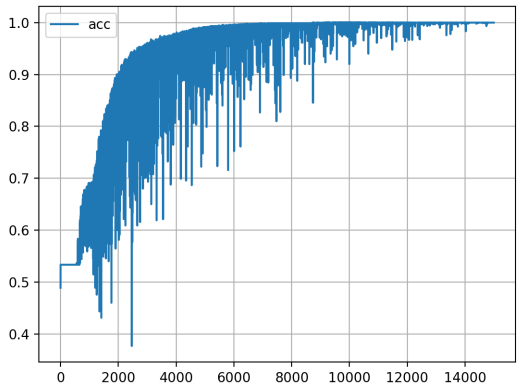
Number of epochs = 8000
Testing accuracy =92 %



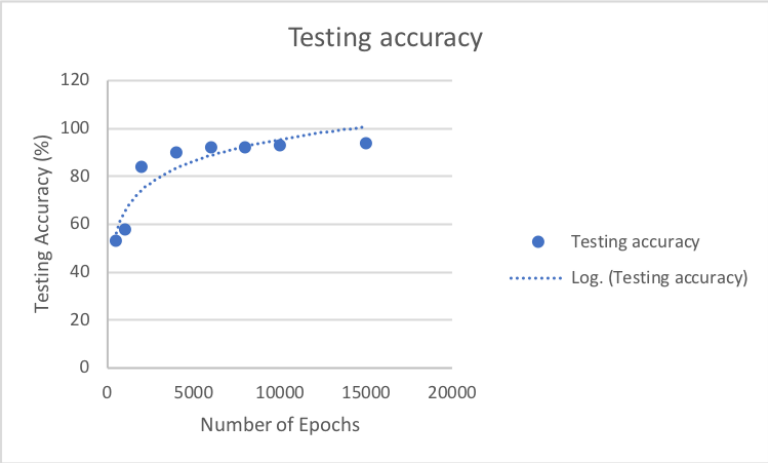
Number of epochs = 10000
Testing accuracy = 93%



Number of epochs = 15000
Testing accuracy =94%



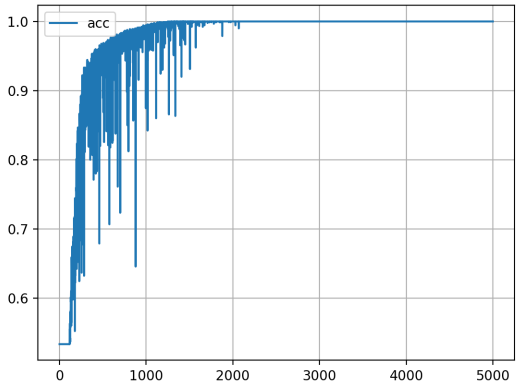
2.3.1 Conclusion



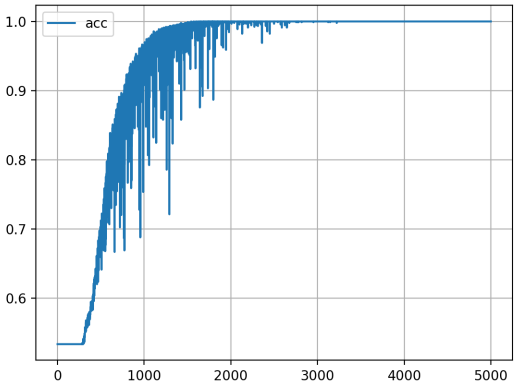
As shown in graphic above, the increase between testing accuracy and number of Epochs follows more in Logarithmic instead of in linear. It mainly because testing accuracy has a low increase at low amount of number of epochs. The accuracy increase gradually after it reaches 93%. When there are woo many epochs, it will also occur overfitting.

2.4 Batch Size

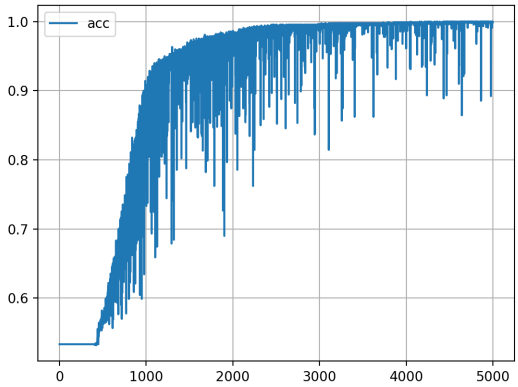
Batch size = 16
Testing accuracy = 88%



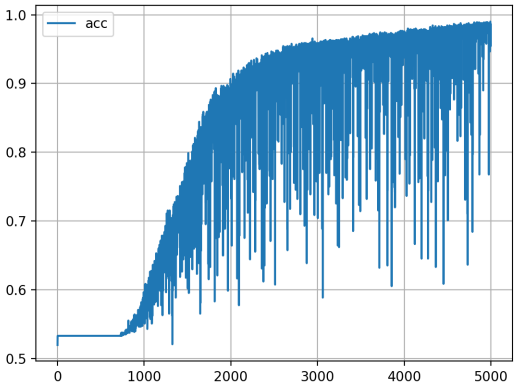
Batch size = 32
Testing accuracy = 84%



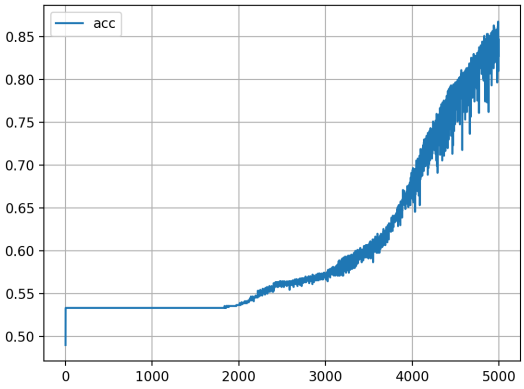
Batch size = 64
Testing accuracy = 96%



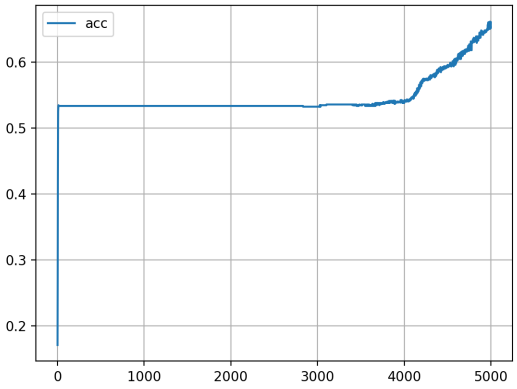
Batch size = 128
Testing accuracy =90 %



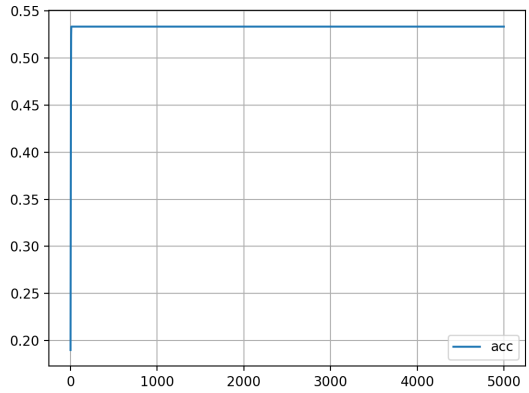
Batch size = 256
Testing accuracy = 86%



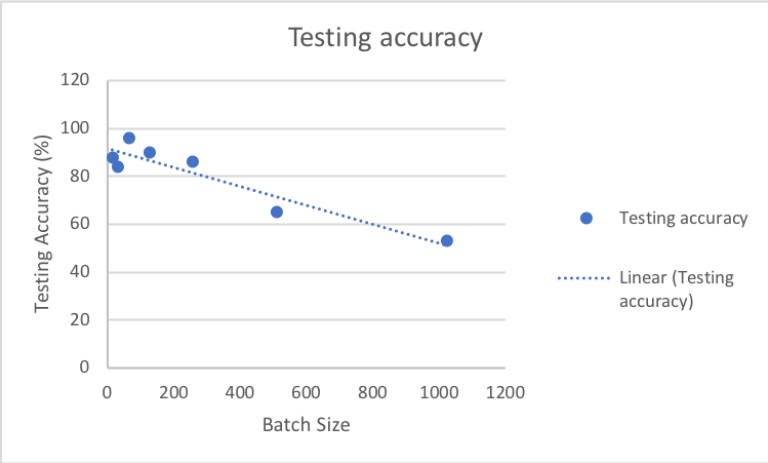
Batch size = 512
Testing accuracy = 65%



Batch size = 1024
Testing accuracy = 53%



2.4.1 Conclusion



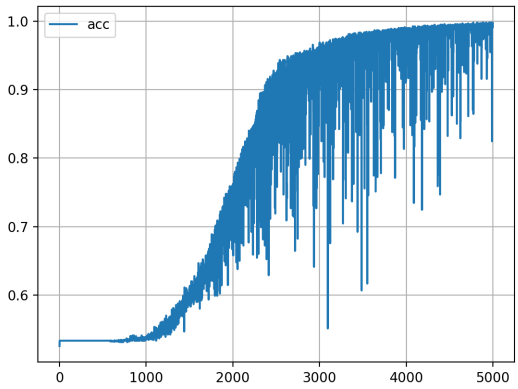
Batch size decides the amount of task being processed at a time. In our case, the testing accuracy tends to increase at the low batch size. But in general, it will decrease accuracy with batch size increase. There may be 2 answer to this situation. Since Batch size will take advantage of power GPU, my first guess is it can be my GPU of the computer computing this result is not powerful enough. On the other hand, using batch size will influence the training time, the error that I achieve, the gradient shifts.

2.5 Activation Funtions

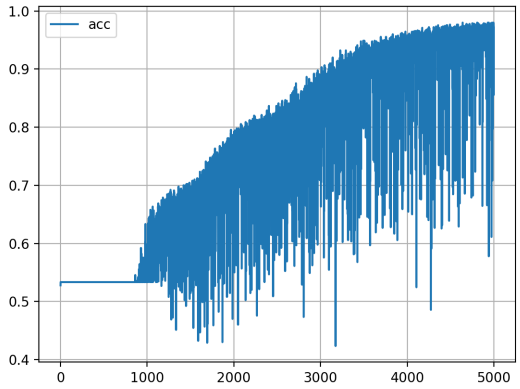
In this section I will study the difference in testing accuracy with different activation functions. All the testing accuracy will be computed under the default hyper-parameter:

number of hidden neurons layer 1 = 100
Learning rate = 0.05
number of epochs = 5000
batch size = 128

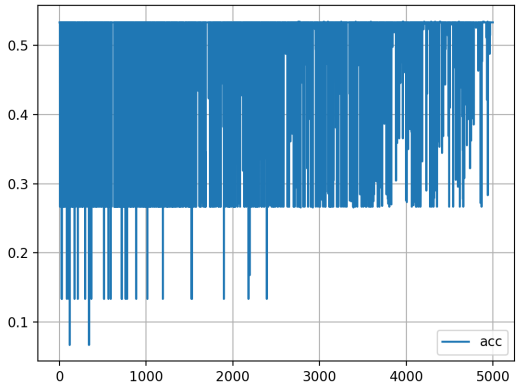
Relu
Testing accuracy = 92%



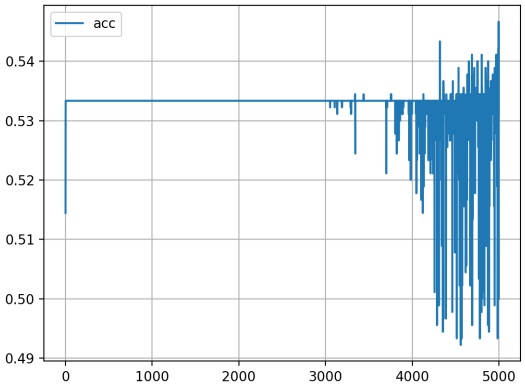
Leaky relu
Testing accuracy = 89%



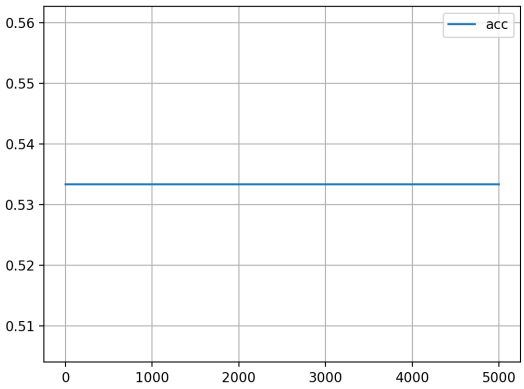
Softplus
Testing accuracy = 53%



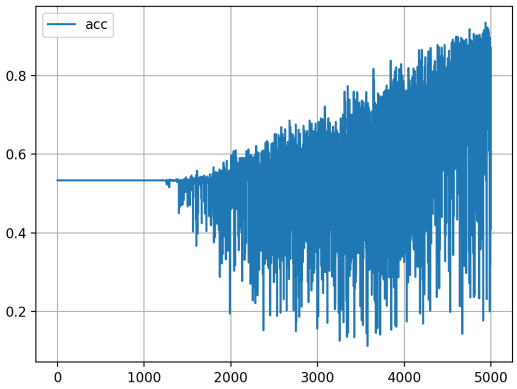
Softsign
Testing accuracy =47 %



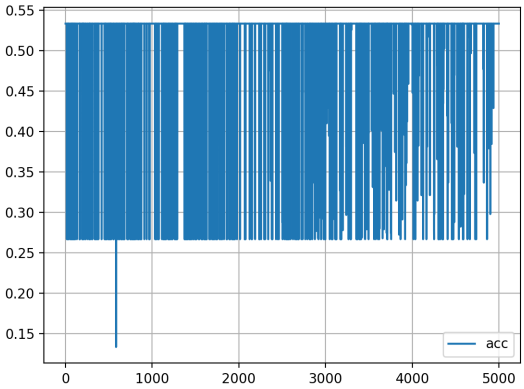
Elu
Testing accuracy = 53%



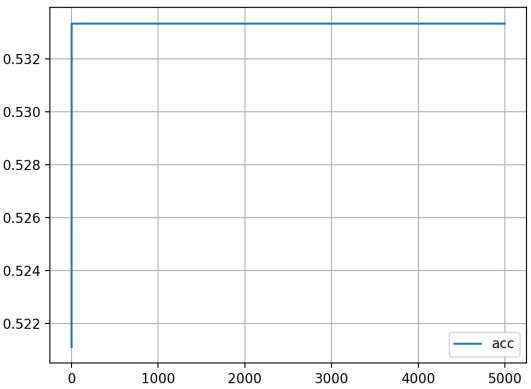
Selu
Testing accuracy = 74%



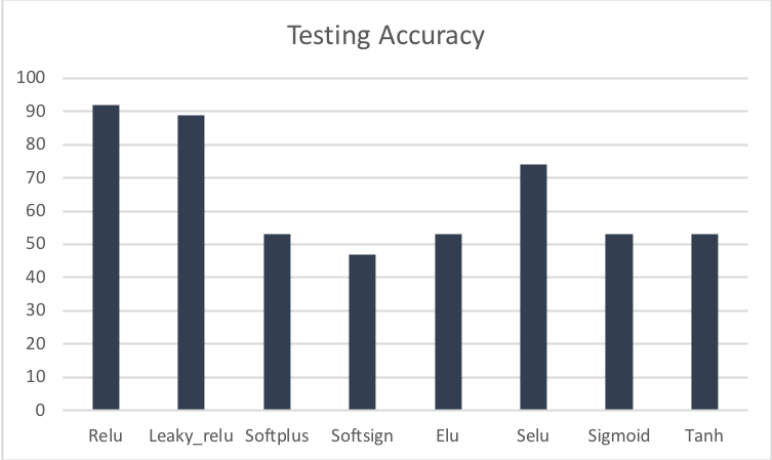
Sigmoid
Testing accuracy = 53%



Selu
Testing accuracy = 53%



2.5.1 Conclusion



Activation function are feed-forward nets that help us deal with nonlinear problems. Since we are using less than 3 layer in solving FizzBuff, all the activation function tends to similar accuracy while relu and leaky_relu have higher accuracy.

3 Summary

By studying hyper-parameter and activation functions, the Software 2.0 has great potential in dealing with large tasks and complicated nonlinear problem solving. While Software 1.0 can achieve 100% accuracy by using standard logic, it can only be used in single layer and linear problem solving. By adjusting activation function to fit the different problem environment and setting the right hyper-parameter, Software 2.0 by using tensor-flow can reach a great accuracy.