Project 1.1: Software 1.0 Versus Software 2.0

Kiran Prabhakar
Department of Computer Science
University at Buffalo
Buffalo, NY 14260
kprabhak@buffalo.edu

Abstract

The project compares the two problem solving approaches by considering the problem called FizzBuzz. The purpose of the project is to measure the accuracy between the logic based approach and the machine learning approach.

1. Software 1.0:

The standard version of the software uses the if-else statements and basic divisibility logic to solve the problem with 100% efficiency.

2. Software 2.0:

The software to solve the FizzBuzz problem using the machine learning approach leverages the open source neural network library called **Keras,** which runs on top of the Tensorflow. The report explains about the various hyper parameters and also compares the accuracy of the software for different hyper parameters**.**

3. Hyper Parameters:

Hyper parameters are the various inputs considered while designing a model, which has significant relevance in determining the accuracy. The below hyper parameters and definitions are to be taken into consideration while developing the model.

3.1 Activation function: It determines whether a node in the network should fire or not. It takes the weighted sum of inputs from various nodes and determines the relevance of current node in the network. The various activation functions that are popularly used are as below.

Name	Range	Definition
Linear	(-infinity, infinity)	A(x) = x
Binary Step	0 or 1	A(x) = 0 if x<0; = 1 if x>1
Sigmoid	(0,1)	$A = \frac{1}{1+e^{-x}}$
Tanh	(-1,1)	2*Sigmoid(2x) -1
Relu	[0, infinity)	A(x) = x; if x > 0 = 0; if x < 0

3.2 Epoch: The number of times the training is performed with the entire training set before attaining an accuracy is called Epoch.

3.3 Over fitting: Over fitting is a problem where the model fits too exactly to a particular training set and may fail to provide accurate results for different training sets.

3.4 Drop out factor: In a neural network, to reduce the problem of overfitting, the number of neurons is reduced by a factor called Drop out factor.

3.5 Early Stopping: Early stopping is a technique, where we stop training when a monitored quantity has stopped growing. If early stopping is not performed, the system continues to learn for the provided number of epochs which can lead to overfitting problem.

3.6 Early Patience: The number of epochs with no improvement after which the training is stopped.

3.7 Optimizer: Optimizer is a mathematical function which is used to minimize the error function. The weights and bias are learned and updated by the optimizer function which plays major role in training process.

3.8 Loss Function: Loss functions calculates the difference between the obtained result and expected output. For most classification problem, categorical cross entropy function is used, which requires the targets to be categorized and it provides the output as binary vectors.

3.9 Global minimum: The point where the loss function has minimum value when compared to any other point.

3.10 Local minimum: The point where the function attains minimum value when compared to other points in its neighborhood. However, there are chances that some other point(s) in the same domain can produce a minimum value than the current minimum.

3.11 Gradient Descent: It is a technique used for determining minimum value of a loss function by progressively moving towards global minimum by moving along the decreasing gradient and with learning rate factor.

4. Analysis of model creation for solving Fizzbuzz problem:

Case-1:

Initially consider a system with **no activation function**, which uses the default linear activation function and trains the system for 1000 Epochs with below hyper parameters.

Table 4.1: Hyper Parameters with no activation

Drop Out	First Layer	Batch Size	Activation	Optimizer	Epochs	Early Stopping Allowed
0.2	512	256	No Activation	Rmsprop	1000	No

This resultant accuracy of the model is shown as below graph.

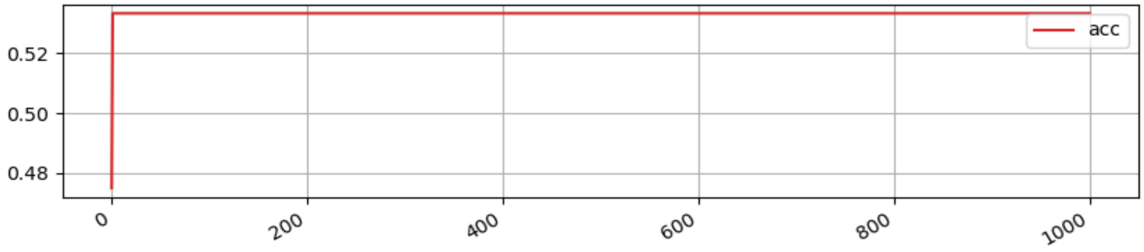


Figure 4.1: Accuracy – 53%

For the very first epoch the system attains 53 % accuracy and remains constant across epochs. The accuracy doesn't change even when any other hyper parameter changes. This is because of the nature of the activation function **y = cx**; for any real constant 'c'. As the derivate of the function is constant, the gradient has no relation with the input value. So the model doesn't learn even if the number of epochs increases.

Case –2:

Consider a different activation function called **tanh** and other parameters which are same as considered in previous case. The parameters are as shown below.

Table 4.2: Hyper Parameters with tanh activation

Drop Out	First Layer	Batch Size	Activation	Optimizer	Epochs	Early Stopping Allowed
0.2	512	256	Tanh	Rmsprop	1000	No

This gives an output with 53 % accuracy which didn't increase the accuracy. The model would have gained accuracy if the number of epochs is increased. But, it will lead to problem of overfitting.

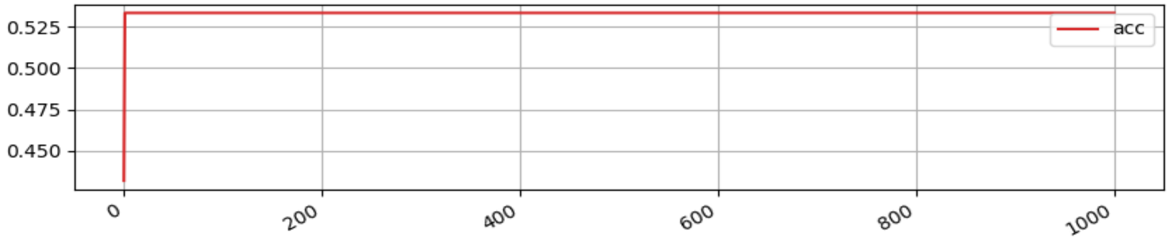


Figure 4.2: Accuracy – 53%

Case-3:

In the third scenario, consider **sigmoid** as the activation function and set the number of epochs to 10000 without early stopping. The output of the activation function is in the range (0,1). This will give a graph, where the average accuracy increases to 81% along with increase in epochs. The below mentioned table gives the details of the hyper parameters for the graph.

Table 4.3: Hyper Parameters with sigmoid activation

Drop Out	First Layer	Batch Size	Activation	Optimizer	Epochs	Early Stopping Allowed
0.2	256	256	Sigmoid	Rmsprop	10000	No

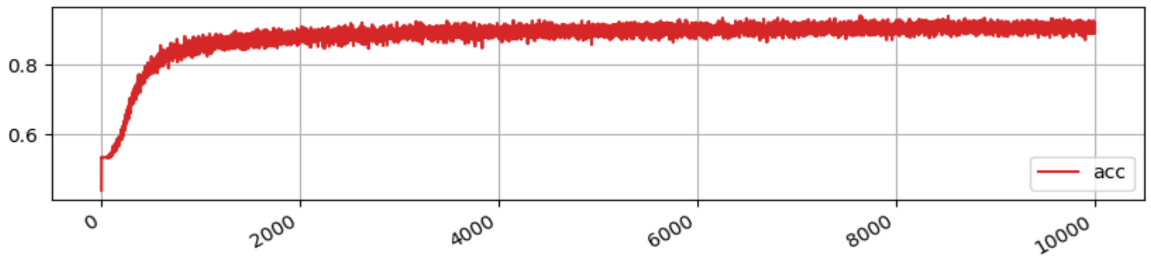


Figure 4.3: Accuracy – 81%

The accuracy obtained with the above activation function is much better when compared to the previous activation function. However, training the model without early stopping is a bad idea, because with increasing epochs the model tries to exactly fit with the data and will lead to overfitting problems. So in the later cases, early stopping is introduced to circumvent the overfitting problem.

Case-4:

In this case, consider **Relu** as the activation function and **Adamax** as optimizer along with the below hyper parameters.

Table 4.4: Hyper Parameters with relu activation

Drop Out	**First Layer**	**Batch Size**	**Activation**	**Optimizer**	**Epochs**	**Early Stopping Allowed**
0.2	512	256	relu	adamax	10000	yes

The accuracy obtained with this activation function after training for 1335 epochs is 95% which is shown in the below table and graph.

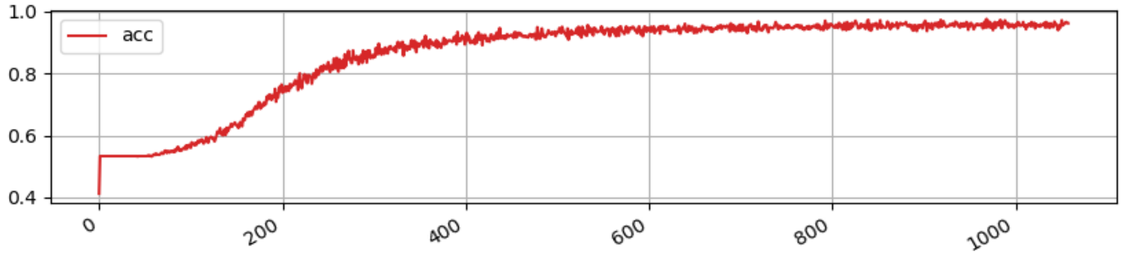
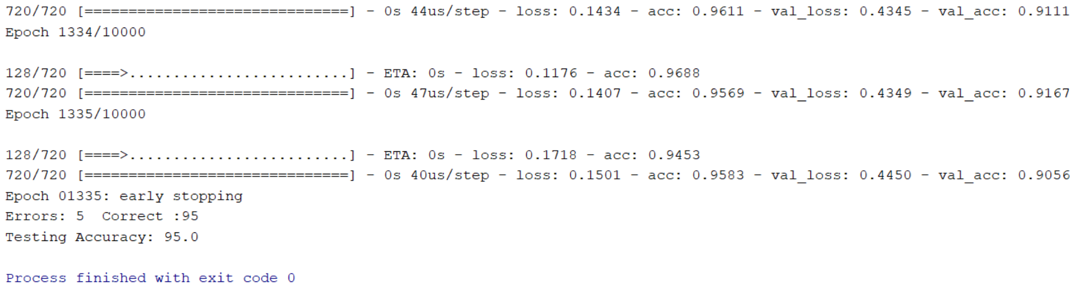


Figure 4.4: Accuracy – 95%



```
720/720 [==============================] - 0s 44us/step - loss: 0.1434 - acc: 0.9611 - val_loss: 0.4345 - val_acc: 0.9111
Epoch 1334/10000

128/720 [====>.........................] - ETA: 0s - loss: 0.1176 - acc: 0.9688
720/720 [==============================] - 0s 47us/step - loss: 0.1407 - acc: 0.9569 - val_loss: 0.4349 - val_acc: 0.9167
Epoch 1335/10000

128/720 [====>.........................] - ETA: 0s - loss: 0.1718 - acc: 0.9453
720/720 [==============================] - 0s 40us/step - loss: 0.1501 - acc: 0.9583 - val_loss: 0.4450 - val_acc: 0.9056
Epoch 01335: early stopping
Errors: 5  Correct :95
Testing Accuracy: 95.0

Process finished with exit code 0
```

Figure 4.5: 1335 epochs with 95% accuracy

Case-5:

Consider making the drop-out rate to zero for the above case, then the potential problem of over fitting occurs, where the model produces 100% accuracy after running for 998 epochs as shown in the below graph and table.

Table 4.5: Hyper Parameters with no drop-out

Drop Out	**First Layer**	**Batch Size**	**Activation**	**Optimizer**	**Epochs**	**Early Stopping Allowed**
0	512	256	Relu	Adamax	10000	Yes

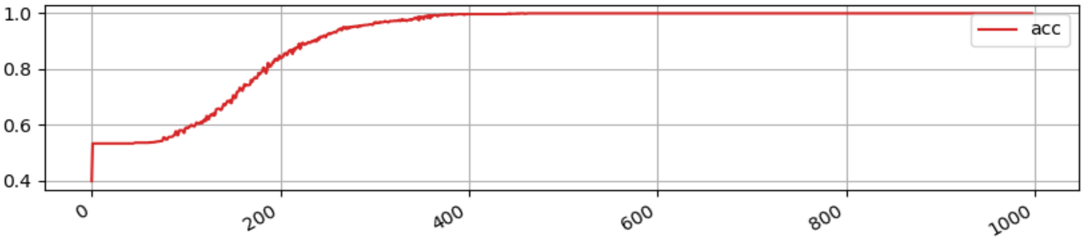
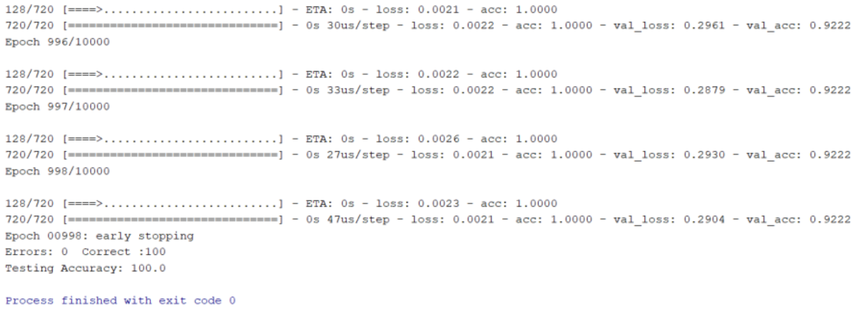


Figure 4.6: 100% accuracy – over fit



```
128/720 [====>.........................] - ETA: 0s - loss: 0.0021 - acc: 1.0000
720/720 [==============================] - 0s 30us/step - loss: 0.0022 - acc: 1.0000 - val_loss: 0.2961 - val_acc: 0.9222
Epoch 996/10000

128/720 [====>.........................] - ETA: 0s - loss: 0.0022 - acc: 1.0000
720/720 [==============================] - 0s 33us/step - loss: 0.0022 - acc: 1.0000 - val_loss: 0.2879 - val_acc: 0.9222
Epoch 997/10000

128/720 [====>.........................] - ETA: 0s - loss: 0.0026 - acc: 1.0000
720/720 [==============================] - 0s 27us/step - loss: 0.0021 - acc: 1.0000 - val_loss: 0.2930 - val_acc: 0.9222
Epoch 998/10000

128/720 [====>.........................] - ETA: 0s - loss: 0.0023 - acc: 1.0000
720/720 [==============================] - 0s 47us/step - loss: 0.0021 - acc: 1.0000 - val_loss: 0.2904 - val_acc: 0.9222
Epoch 00998: early stopping
Errors: 0  Correct :100
Testing Accuracy: 100.0

Process finished with exit code 0
```

Figure 4.7: 1335 epochs with 100% accuracy

With the above hyper parameters, the model appears to have the problem of overfitting. To circumvent the issue, the values for dropout and the epoch count should be properly decided.

Case-6:

All the above mentioned analysis were conducted by having one dense layer in the network. Now, in the final case, one more layer is introduced in the network with same number of nodes as earlier layer and with same values for other hyper parameters as shown below.

Table 4.6: Hyper Parameters with drop-out and two layers

No. of Layers	**Drop Out**	**Nodes in each layer**	**Batch Size**	**Activation**	**Optimizer**	**Epochs**	**Early Stopping Allowed**
2	0.1	512	128	Relu	Adamax	10000	Yes

The accuracy of the system has increased to 98% after running for 500 epochs before early stopping. The result and graph for these inputs is as shown below.

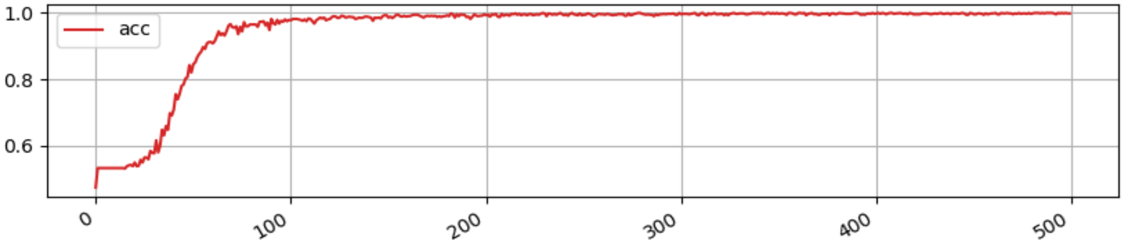
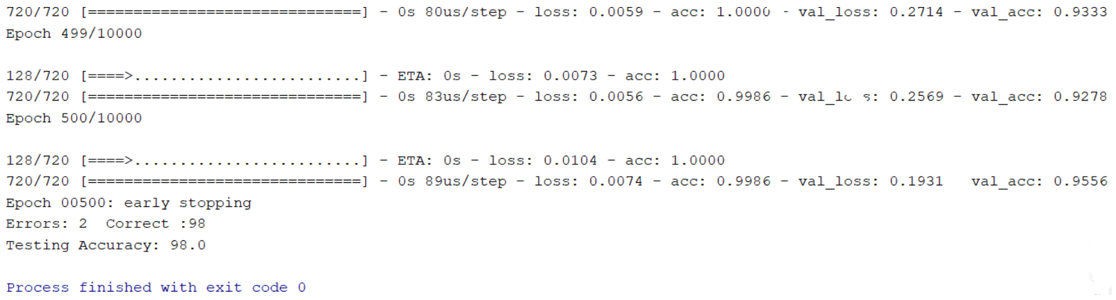


Figure 4.8: 484 epochs with 98% accuracy



```
720/720 [==============================] - 0s 80us/step - loss: 0.0059 - acc: 1.0000 - val_loss: 0.2714 - val_acc: 0.9333
Epoch 499/10000

128/720 [====>.........................] - ETA: 0s - loss: 0.0073 - acc: 1.0000
720/720 [==============================] - 0s 83us/step - loss: 0.0056 - acc: 0.9986 - val_loss: 0.2569 - val_acc: 0.9278
Epoch 500/10000

128/720 [====>.........................] - ETA: 0s - loss: 0.0104 - acc: 1.0000
720/720 [==============================] - 0s 89us/step - loss: 0.0074 - acc: 0.9986 - val_loss: 0.1931   val_acc: 0.9556
Epoch 00500: early stopping
Errors: 2  Correct :98
Testing Accuracy: 98.0

Process finished with exit code 0
```

Figure 4.8: 500 epochs with 98% accuracy with two layers

5. Conclusion:

Thus, by tweaking the values for various hyper parameters, Software 2.0 provides different accuracy for different inputs. The best accuracy for FizzBuzz problem is obtained without any overfitting with the below combination of hyper parameters.

No. of Layers	Drop Out	Nodes in each layer	Batch Size	Activation	Optimizer	Epochs	Early Stopping Allowed	Accuracy
2	0.1	512	128	Relu	Adamax	10000	Yes	98

6. References:

- https://keras.io
- https://www.tensorflow.org/tutorials
- https://medium.com/data-science-group-iitr
- https://towardsdatascience.com