CSE 574 LEC001

Project 1.1: FizzBuzz

Intro To Machine Learning

The FizzBuzz Game

1, 2, Fizz!, 4, Buzz!, Fizz!, 7,
8, Fizz!, Buzz!, 11, Fizz!, 13,
14, FizzBuzz!, 16, 17, Fizz!...

Malini Anbazhagan

Person# 50289383

University at Buffalo

Buffalo, NY 14260

Contents

Introduction

Fizzbuzz is a math game where one has to say the numbers 1,2, “Fizz” instead of 3 or 3’s multiple, 4, “Buzz” instead of 5 or 5’s multiple,”Fizz”, 7, 8, ”Fizz” , ”Buzz”,11, “Fizz”,13,14, ”FizzBuzz” instead of 15 or 15’s multiple.

So it goes something like this 1, 2 , Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, FizzBuzz, 16, 17, Fizz, 19, Buzz, Fizz, 22, 23……………………………….

The project Fizzbuzz is a simple project which is divided in two parts.

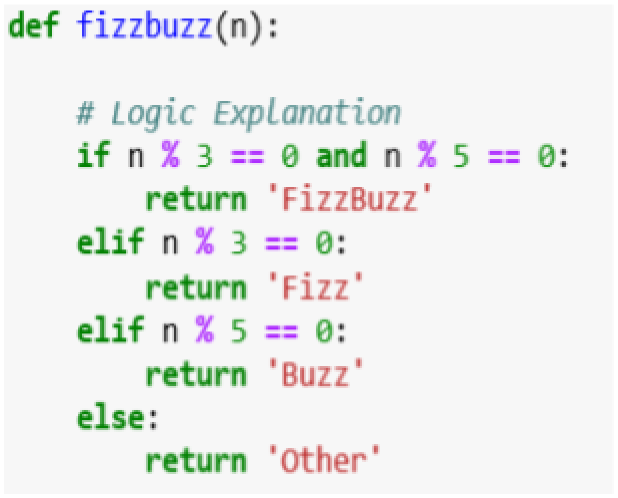
- Software 1.0
- Software 2.0

Software 1.0 is a simple python program for the fizzbuzz.

Software 2.0 is a machine learning code which again is python code which required many packages such as keras, tensorflow, pandas,matlib etc.. The code makes the machine learn “Fizzbuzz”. The machine basically learns and tries to predict the outcome.

Code Explanation

Software 1.0:



```python
def fizzbuzz(n):

    # Logic Explanation
    if n % 3 == 0 and n % 5 == 0:
        return 'FizzBuzz'
    elif n % 3 == 0:
        return 'Fizz'
    elif n % 5 == 0:
        return 'Buzz'
    else:
        return 'Other'
```

Figure 2.1

A function names fizzbuzz is defined where the argument is the number.

For the multiple of 15's: If the number is divisible by 3 and 5 return "FizzBuzz".

For the multiple of 3's: If the number is divisible

by 3 return "Fizz".

For the multiple of 5's: If the number is divisible by 5 return "Buzz".

For other cases: If the number is not divisible by 3 and 5 return "Other"

Software:2.0:

Training and testing dataset set in created in CSV format. From training set, the model can be built upon and from testing set is used for evaluating the model or validate the model.

The data is processed. The labels are added in this function where label are "Fizz", "Buzz", "FizzBuzz" and "Other"

Then a model is defined. With the help of keras and tensorflow. Using the functional API given some input tensors and output tensors, one can instantiate a model. Some method classes such as dense is used to know how dense the network is. [1] Hyperparameters are the variables which determines the network structure and the variables which determine how the network is trained. [2] For example, number of nodes, number of layers, drop out, epochs, batch size, iterations etc [3].

Here is some example to understand few of the hyperparameters.

We can divide the dataset of 4000 examples into batches of 400 then it will take 10 iterations to complete 1 epoch. So batch size= 400, iterations=10, epoch =1.

Drop out is a regularization technique to avoid overfitting.

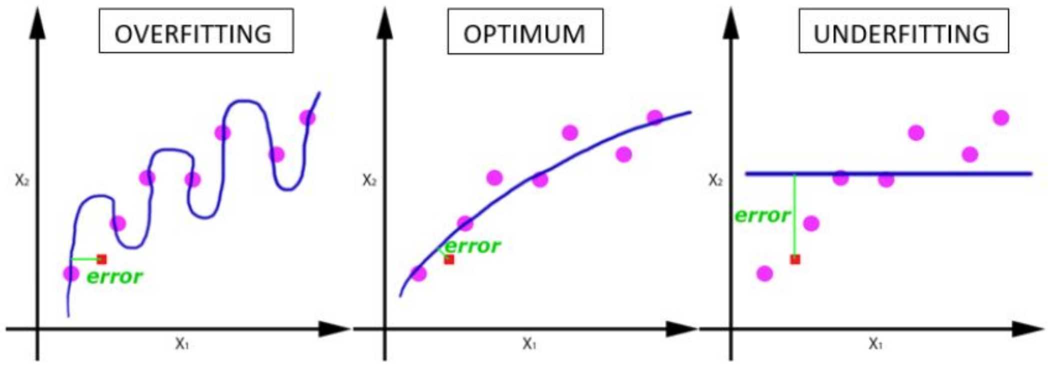


Figure 2.2

Hyperparameters are set before training the model.

After making changes in the code by trial and error method, I achieved **91% accuracy** in the given code. However, for every run the accuracy value changes due to drop outs.

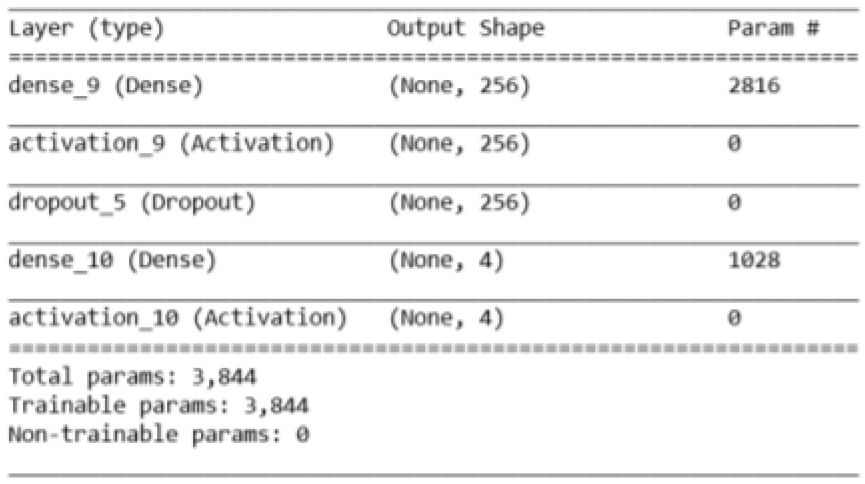
The values were changed in the hyperparameters and tested by trial and error method.

The rest of runs are shown below.

The first table shows change in drop_outs. The changes are noted in the table and graphs.

Run 1
## Model Definition
input_size = 10
drop_out = 0.2
first_dense_layer_nodes = 256
second_dense_layer_nodes = 4

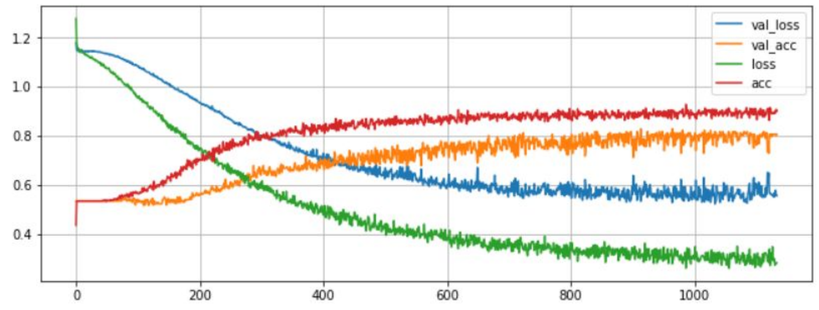
#Run model
validation_data_split = 0.2
num_epochs = 10000
model_batch_size = 128
tb_batch_size = 32
early_patience = 100



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 256)               2816
_________________________________________________________________
activation_9 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_10 (Dense)             (None, 4)                 1028
_________________________________________________________________
activation_10 (Activation)   (None, 4)                 0
=================================================================
Total params: 3,844
Trainable params: 3,844
Non-trainable params: 0
_________________________________________________________________
```

```
%matplotlib inline
df = pd.DataFrame(history.history)
df.plot(grid=True, figsize=(11,4))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24688fdf668>
```



Errors: 15 Correct :85 Testing Accuracy: 85.0

Run 2
## Model Definition
input_size = 10
drop_out = 0.19
first_dense_layer_nodes = 256
second_dense_layer_nodes = 4
#Run model
validation_data_split = 0.2
num_epochs = 10000
model_batch_size = 128
tb_batch_size = 32
early_patience = 100

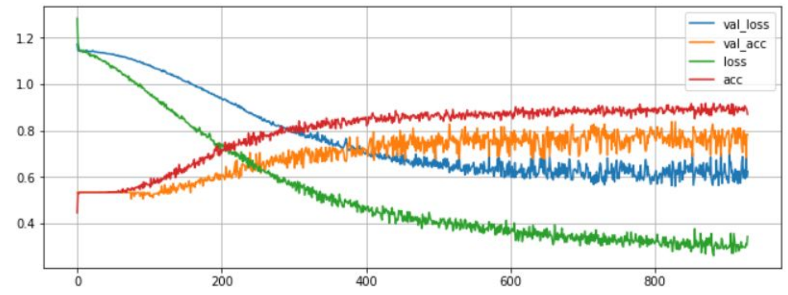
(This is gave me the best accuracy of 91 % accuracy)

```
model = get_model()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_17 (Dense)             (None, 256)               2816
_________________________________________________________________
activation_17 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_9 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_18 (Dense)             (None, 4)                 1028
_________________________________________________________________
activation_18 (Activation)   (None, 4)                 0
=================================================================
Total params: 3,844
Trainable params: 3,844
Non-trainable params: 0
```

```
%matplotlib inline
df = pd.DataFrame(history.history)
df.plot(grid=True, figsize=(11,4))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2468b393940>
```



Errors: 9 Correct :91 Testing Accuracy: 91.0

Run 3
## Model Definition
input_size = 10
drop_out = 0.18
first_dense_layer_nodes = 256
second_dense_layer_nodes = 4

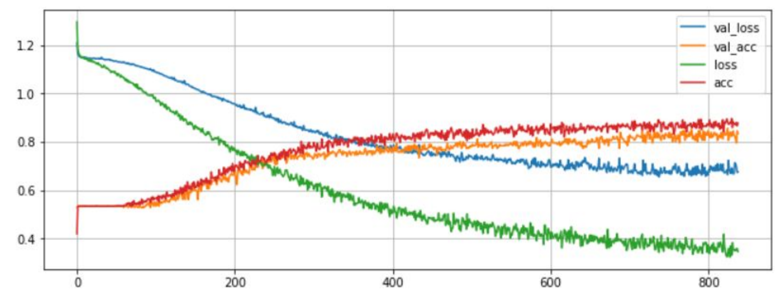
#Run model
validation_data_split = 0.2
num_epochs = 10000
model_batch_size = 128
tb_batch_size = 32
early_patience = 100

```
model = get_model()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_15 (Dense)             (None, 256)               2816
_________________________________________________________________
activation_15 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_8 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_16 (Dense)             (None, 4)                 1028
_________________________________________________________________
activation_16 (Activation)   (None, 4)                 0
=================================================================
Total params: 3,844
Trainable params: 3,844
Non-trainable params: 0
_________________________________________________________________
```

```
%matplotlib inline
df = pd.DataFrame(history.history)
df.plot(grid=True, figsize=(11,4))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x24689d278d0>
```



Errors: 11 Correct :89 Testing Accuracy: 89.0

Run 4

## Model Definition
input_size = 10
drop_out = 0.195
first_dense_layer_nodes = 256
second_dense_layer_nodes = 4

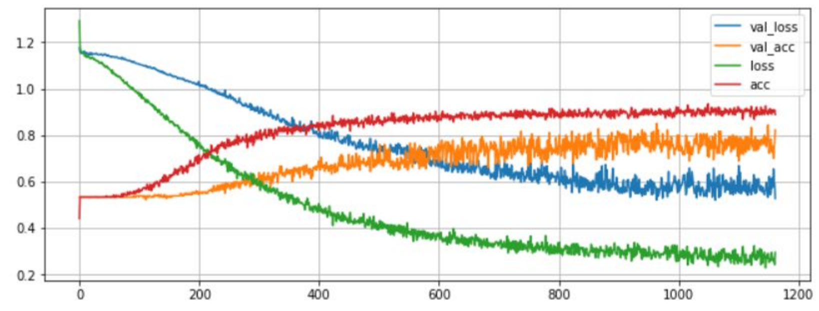
#Run model
validation_data_split = 0.2
num_epochs = 10000
model_batch_size = 128
tb_batch_size = 32
early_patience = 100

```
model = get_model()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_21 (Dense)             (None, 256)               2816
_________________________________________________________________
activation_21 (Activation)   (None, 256)               0
_________________________________________________________________
dropout_11 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_22 (Dense)             (None, 4)                 1028
_________________________________________________________________
activation_22 (Activation)   (None, 4)                 0
=================================================================
Total params: 3,844
Trainable params: 3,844
Non-trainable params: 0
```

```
%matplotlib inline
df = pd.DataFrame(history.history)
df.plot(grid=True, figsize=(11,4))
```

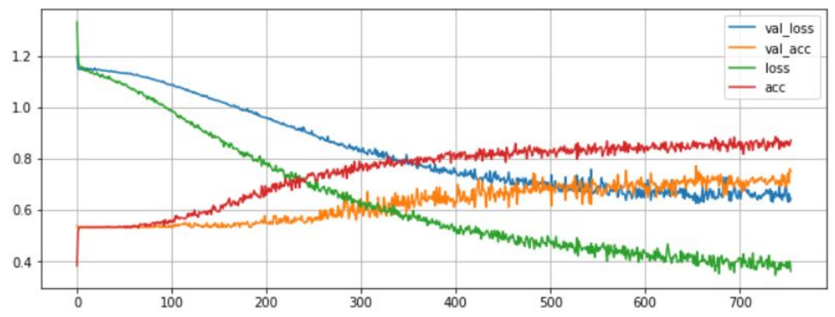
```
<matplotlib.axes._subplots.AxesSubplot at 0x2468c162ef0>
```

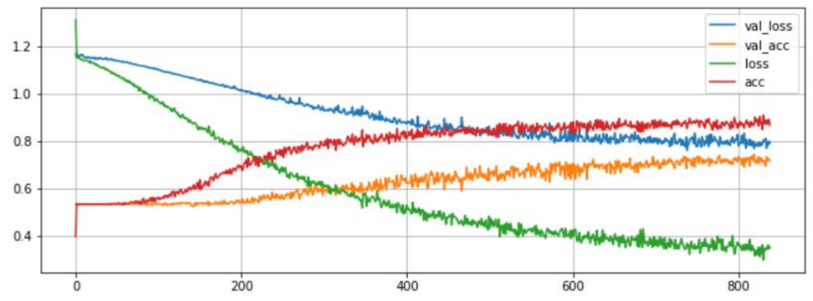


Errors: 13 Correct :87 Testing Accuracy: 87.0

Table 2.1

In the following table, The changes in epochs are noted.

#Run model
validation_data_split = 0.2
num_epochs = 5000
model_batch_size = 128
tb_batch_size = 32
early_patience = 100
Epoch 5000
	
	Errors: 13 Correct :86 Testing Accuracy: 86.0

#Run model
validation_data_split = 0.2
num_epochs = 9000
model_batch_size = 128
tb_batch_size = 32
early_patience = 100
	
	Errors: 9 Correct :91 Testing Accuracy: 91.0

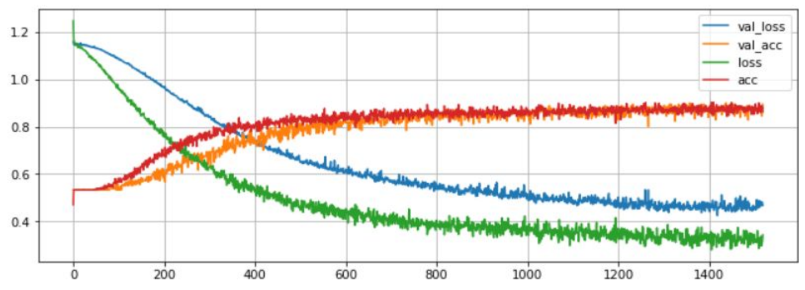
#Run model
validation_data_split = 0.2
num_epochs = 9500
model_batch_size = 128
tb_batch_size = 32
early_patience = 100
	
	Errors: 12 Correct :88 Testing Accuracy: 88.0

Table 2.2

In the code, the accuracy as well as output is displayed.

Summary

From this project I learned various types of libraries, packages, functions, parameters and hyperparameters,API’s available for machine learning. How each of them interacts with the data and delivers the output. I also learned how a model is built and how to train a model.

References

1. Keras documentation https://keras.io/models/model/
2. What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network? By Pranoy Radhakrishnan https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a
3. Epoch vs Batch Size vs Iterations by Sagar Sharma https://towardsdatascience.com/epoch-vs-iterations-vs-batch-size-4dfb9c7ce9c9