Drone & Original Network Implementation

In this task I have implemented the Original Network and Drone Network using Keras in Python. I have pushed the code on my Github repository https://github.com/BigBang0072/GuruDrone

DataSet Information & Visualization

About Data:

The dataset were downloaded from the github repository of HEPDrone.
The dataset was actually divided into two parts:

1. Signal Data
2. Background Data

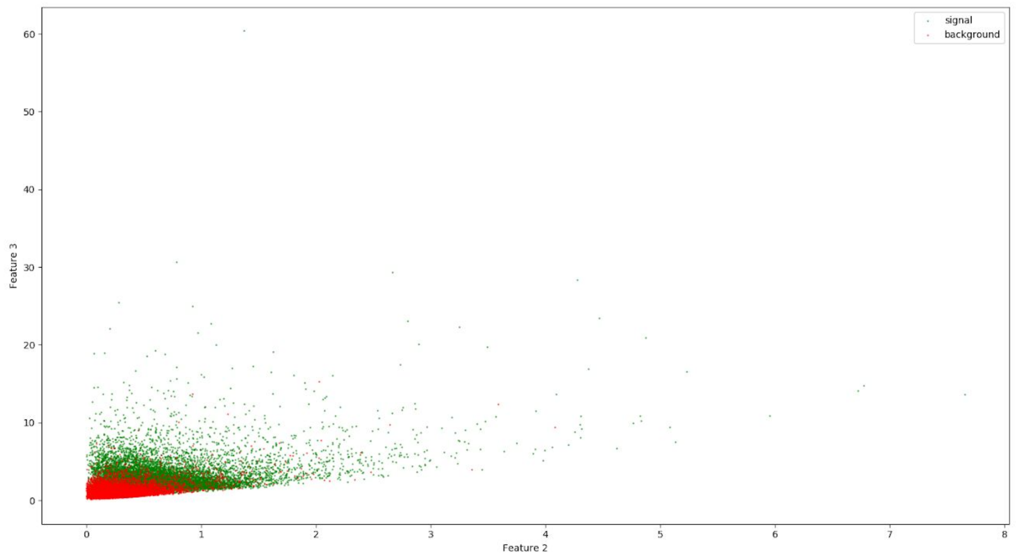
Both the data were saved in pickle file and were retrieved from Sklearn **joblib** library. After retrieving it was converted to Numpy array for all further visualization and analysis
There were 10,000 different examples for each Signal and Background. Each example had 6 entry specifying different features of the data(i.e different parameter of B meson decay(rare event) as Signal and background as D mesons decay)(not familiar with these particle except their definition)

Data Visualization:

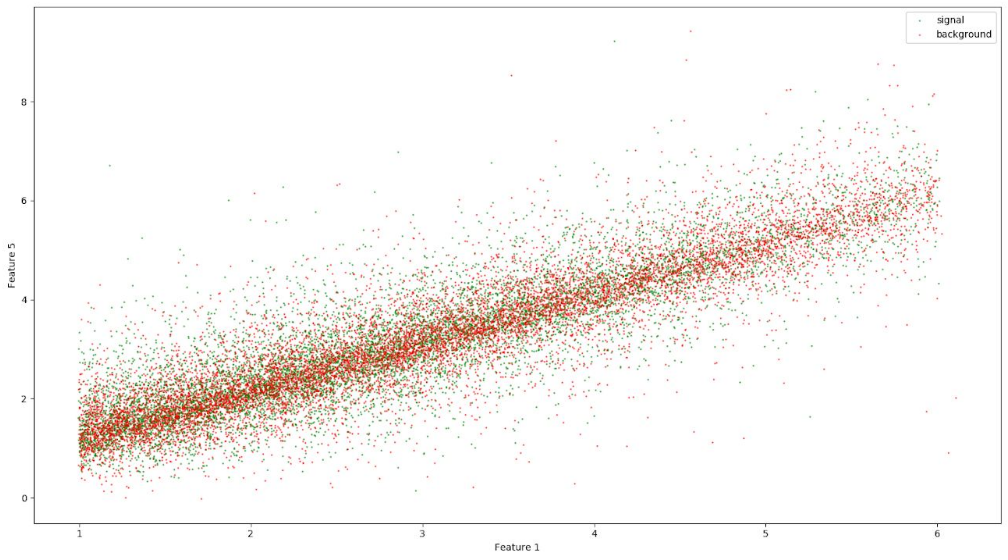
As we were restricted to using just **two variables out of total 6**, proper analysis of data and feature selection were important to see what features could impact the performance of Neural Net and which ones were redundant.

First of all I created a scatter plot of all the Signal and Background data but using only two variables at a time to visually judge which of the pair of the features have clear distinction between the Background and Signal data. Here are few plots showing the Signal(in green) and Background(in red) data with respect to pairwise feature. The full set of the plots could be seen in my repository(in the plot folder described above).
The features are labelled from 0 to 5, ie. first feature is numbered 0 and last is numbered 5.

Feature 2 and Feature 3 and their scatter plots representing Signal among Background.



Feature 1 and feature 5 (here making distinction between signal and background is very difficult)

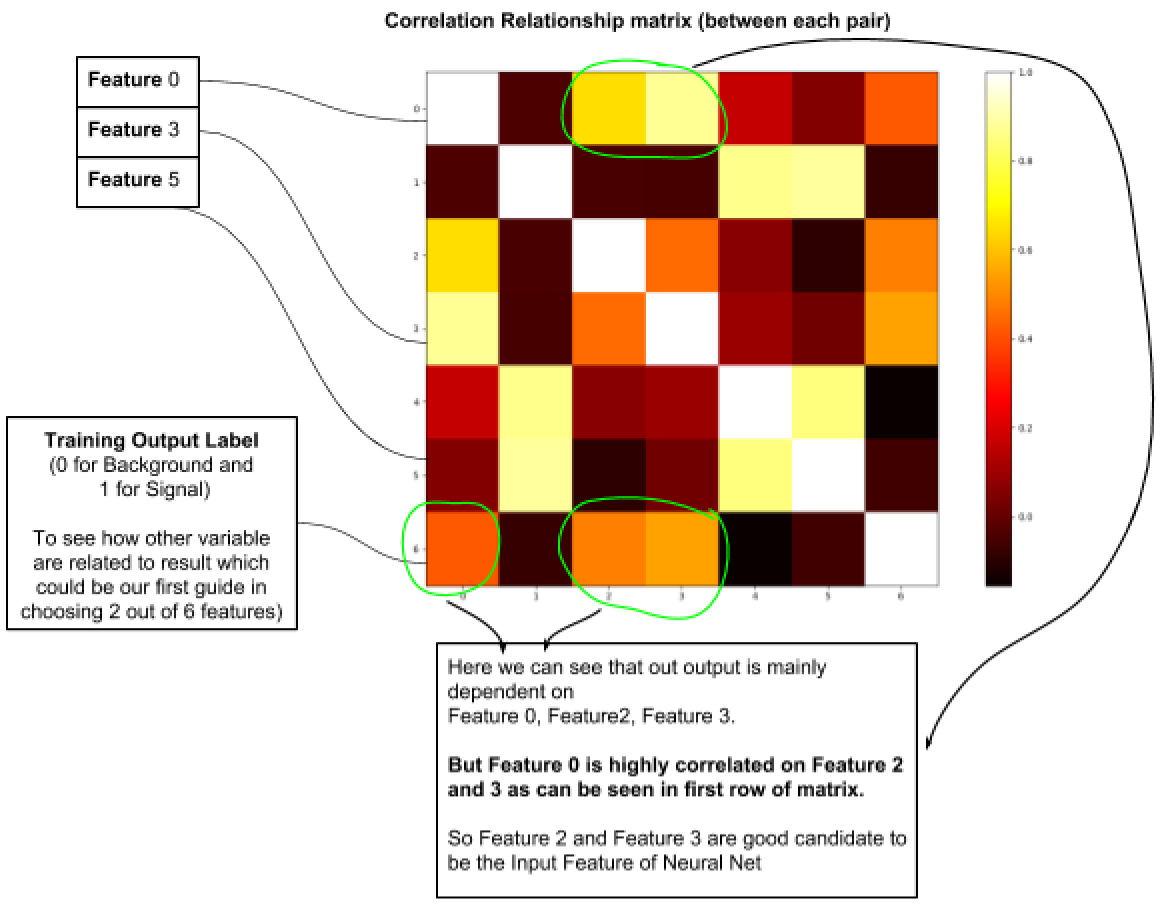


Correlation Plot:

After the visual inspection we needed more concrete proof of importance of different features for classifying (or making) distinction between Signal and background.

So I made a correlation plot, to see the **how variables are related among themselves** and **how variables are related to the output labels** (ie. Background(labelled as 0) and Signal(labelled as 1)).

Correlation value of 1 represent a high degree of “linear” relationship whereas a value of 0 represents no linear relationships, and value of -1 represents a negative linear relationship.



Training Output Label
(0 for Background and
1 for Signal)

To see how other variable are related to result which could be our first guide in choosing 2 out of 6 features)

Here we can see that out output is mainly dependent on
Feature 0, Feature2, Feature 3.

But Feature 0 is highly correlated on Feature 2 and 3 as can be seen in first row of matrix.

So Feature 2 and Feature 3 are good candidate to be the Input Feature of Neural Net

Here the each row represents the correlation of itself with other features. Hence the matrix obtained is **symmetric** (originally, here only the heatmap is plotted for visualization). Also, we could see the white line in primary diagonal showing a variable is having best correlation coefficient with itself.

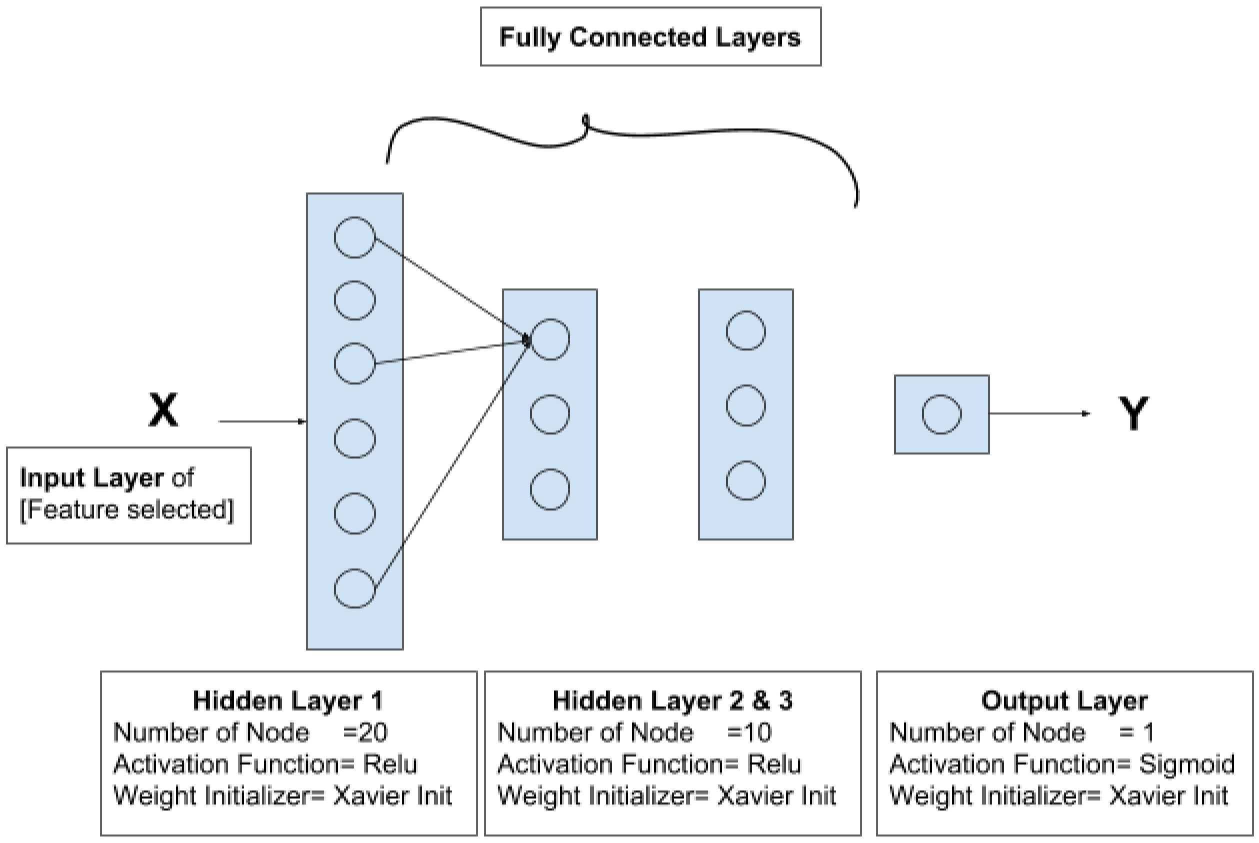
Here the last row represents the dependence of other variable (from 0 to 5) on the output label thus it could give us good way to start feature selection. As we can see the output in greatly dependent on Feature 2 and 3 and those two feature are not much related to each other (sort of “linearly” independent variable as their correlation is closer to zero.) **So, I make a hypothesis that they will be the best two feature set out of all to be a candidate for Network. (This will be proved in next section).**

Original Neural Network

After selecting the 2 feature out of 6, the main task was to test if my intuition was right or not by actually training a simple Feed forward Neural Network.

For uniformity of the results the training epochs and Network Architecture will be kept constant throughout the experiments.

Network architecture:


Fully Connected Layers
X
Y
Input Layer of [Feature selected]
Hidden Layer 1
Number of Node =20
Activation Function= Relu
Weight Initializer= Xavier Init
Hidden Layer 2 & 3
Number of Node =10
Activation Function= Relu
Weight Initializer= Xavier Init
Output Layer
Number of Node = 1
Activation Function= Sigmoid
Weight Initializer= Xavier Init

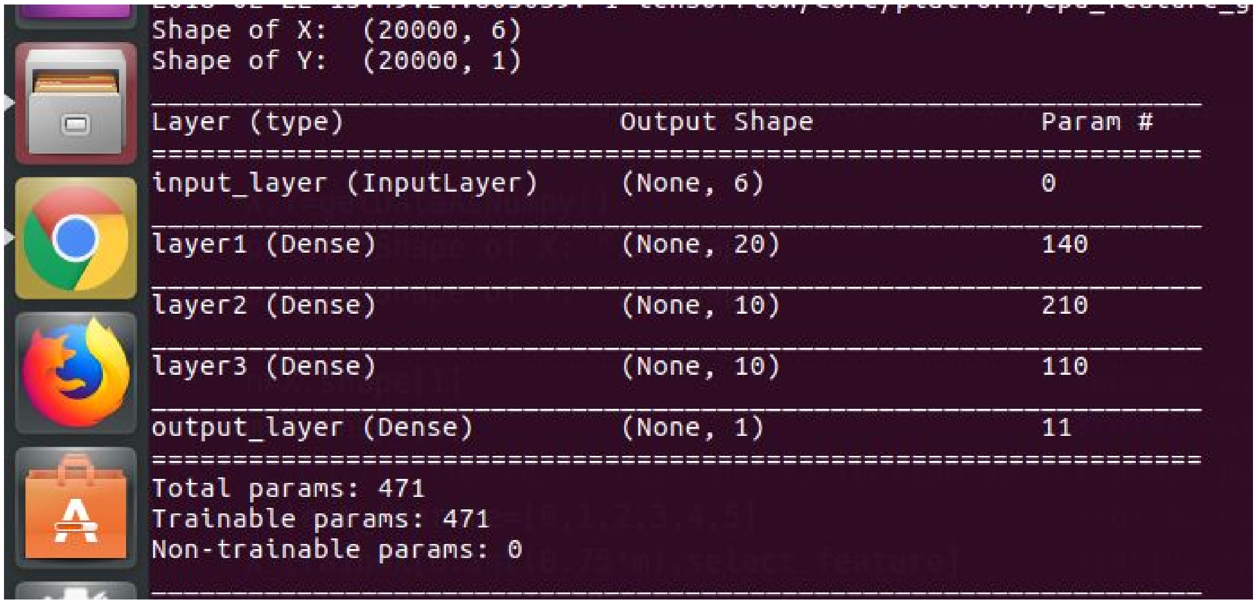
Background Data is Labelled as 0.
Signal Data is Labelled as 1.

Loss Function used	: Binary Cross-Entropy
Number of Training Epoch	: 20(constant)
Evaluation Metric	: Accuracy, Precision, Recall, F1 Score
Train-Test Split	: Train: 75% , Test: 25% (of shuffled dataset)

Test 1: (using all six Features)

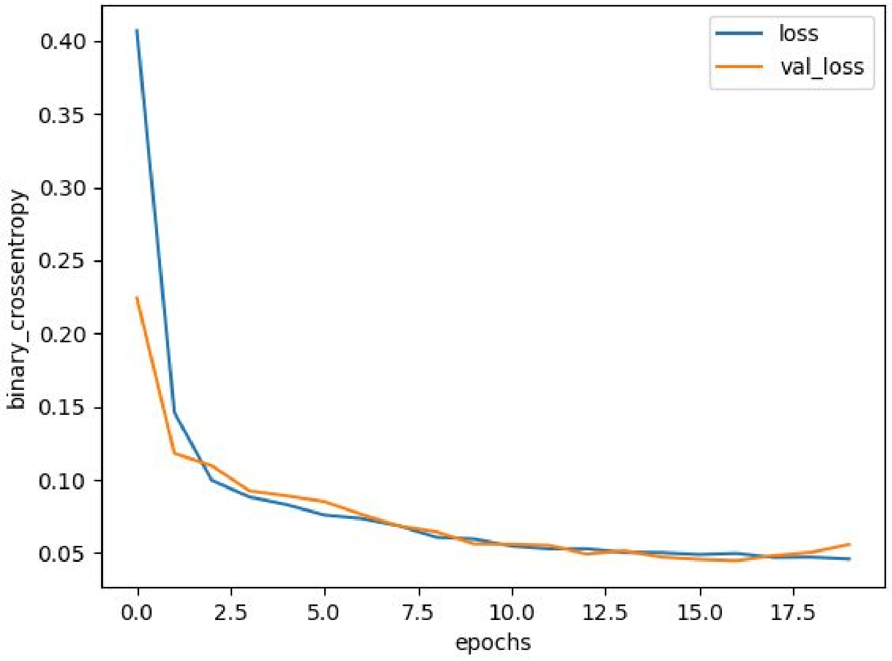
Total number of Parameters : 471

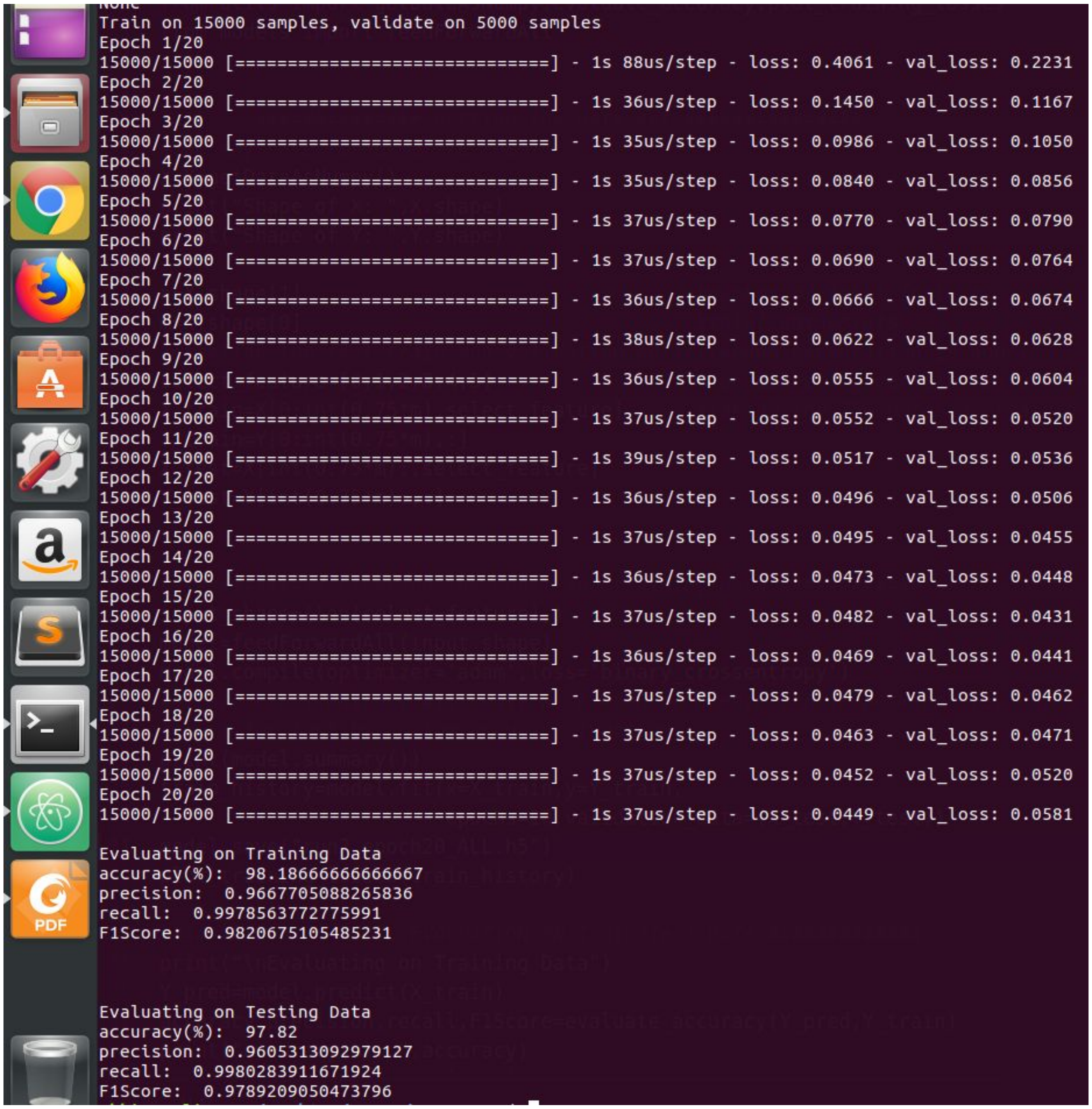
Network Summary (Keras generated) :



```
Shape of X:  (20000, 6)
Shape of Y:  (20000, 1)
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     (None, 6)                 0
_________________________________________________________________
layer1 (Dense)               (None, 20)                140
_________________________________________________________________
layer2 (Dense)               (None, 10)                210
_________________________________________________________________
layer3 (Dense)               (None, 10)                110
_________________________________________________________________
output_layer (Dense)         (None, 1)                 11
=================================================================
Total params: 471
Trainable params: 471
Non-trainable params: 0
_________________________________________________________________
```

Training Loss Curve and Accuracy:



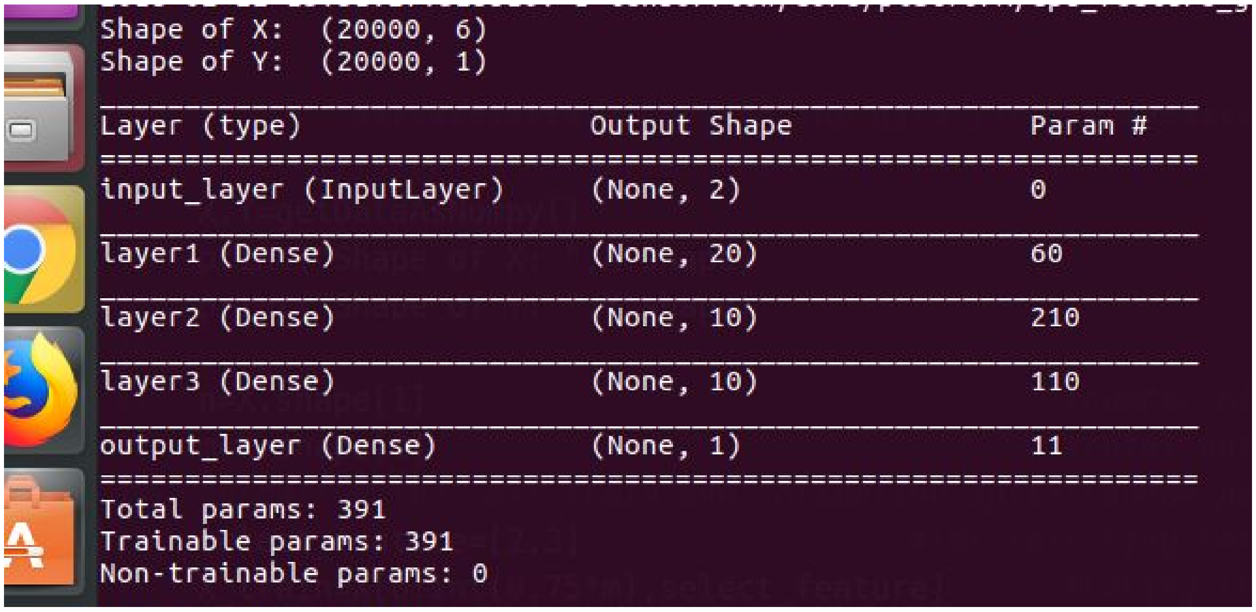


```
Train on 15000 samples, validate on 5000 samples
Epoch 1/20
15000/15000 [==============================] - 1s 88us/step - loss: 0.4061 - val_loss: 0.2231
Epoch 2/20
15000/15000 [==============================] - 1s 36us/step - loss: 0.1450 - val_loss: 0.1167
Epoch 3/20
15000/15000 [==============================] - 1s 35us/step - loss: 0.0986 - val_loss: 0.1050
Epoch 4/20
15000/15000 [==============================] - 1s 35us/step - loss: 0.0840 - val_loss: 0.0856
Epoch 5/20
15000/15000 [==============================] - 1s 37us/step - loss: 0.0770 - val_loss: 0.0790
Epoch 6/20
15000/15000 [==============================] - 1s 37us/step - loss: 0.0690 - val_loss: 0.0764
Epoch 7/20
15000/15000 [==============================] - 1s 36us/step - loss: 0.0666 - val_loss: 0.0674
Epoch 8/20
15000/15000 [==============================] - 1s 38us/step - loss: 0.0622 - val_loss: 0.0628
Epoch 9/20
15000/15000 [==============================] - 1s 36us/step - loss: 0.0555 - val_loss: 0.0604
Epoch 10/20
15000/15000 [==============================] - 1s 37us/step - loss: 0.0552 - val_loss: 0.0520
Epoch 11/20
15000/15000 [==============================] - 1s 39us/step - loss: 0.0517 - val_loss: 0.0536
Epoch 12/20
15000/15000 [==============================] - 1s 36us/step - loss: 0.0496 - val_loss: 0.0506
Epoch 13/20
15000/15000 [==============================] - 1s 37us/step - loss: 0.0495 - val_loss: 0.0455
Epoch 14/20
15000/15000 [==============================] - 1s 36us/step - loss: 0.0473 - val_loss: 0.0448
Epoch 15/20
15000/15000 [==============================] - 1s 37us/step - loss: 0.0482 - val_loss: 0.0431
Epoch 16/20
15000/15000 [==============================] - 1s 36us/step - loss: 0.0469 - val_loss: 0.0441
Epoch 17/20
15000/15000 [==============================] - 1s 37us/step - loss: 0.0479 - val_loss: 0.0462
Epoch 18/20
15000/15000 [==============================] - 1s 37us/step - loss: 0.0463 - val_loss: 0.0471
Epoch 19/20
15000/15000 [==============================] - 1s 37us/step - loss: 0.0452 - val_loss: 0.0520
Epoch 20/20
15000/15000 [==============================] - 1s 37us/step - loss: 0.0449 - val_loss: 0.0581

Evaluating on Training Data
accuracy(%):  98.18666666666667
precision:  0.9667705088265836
recall:  0.9978563772775991
F1Score:  0.9820675105485231


Evaluating on Testing Data
accuracy(%):  97.82
precision:  0.9605313092979127
recall:  0.9980283911671924
F1Score:  0.9789209050473796
```

Test 2: (using selected Features 2 & 3)

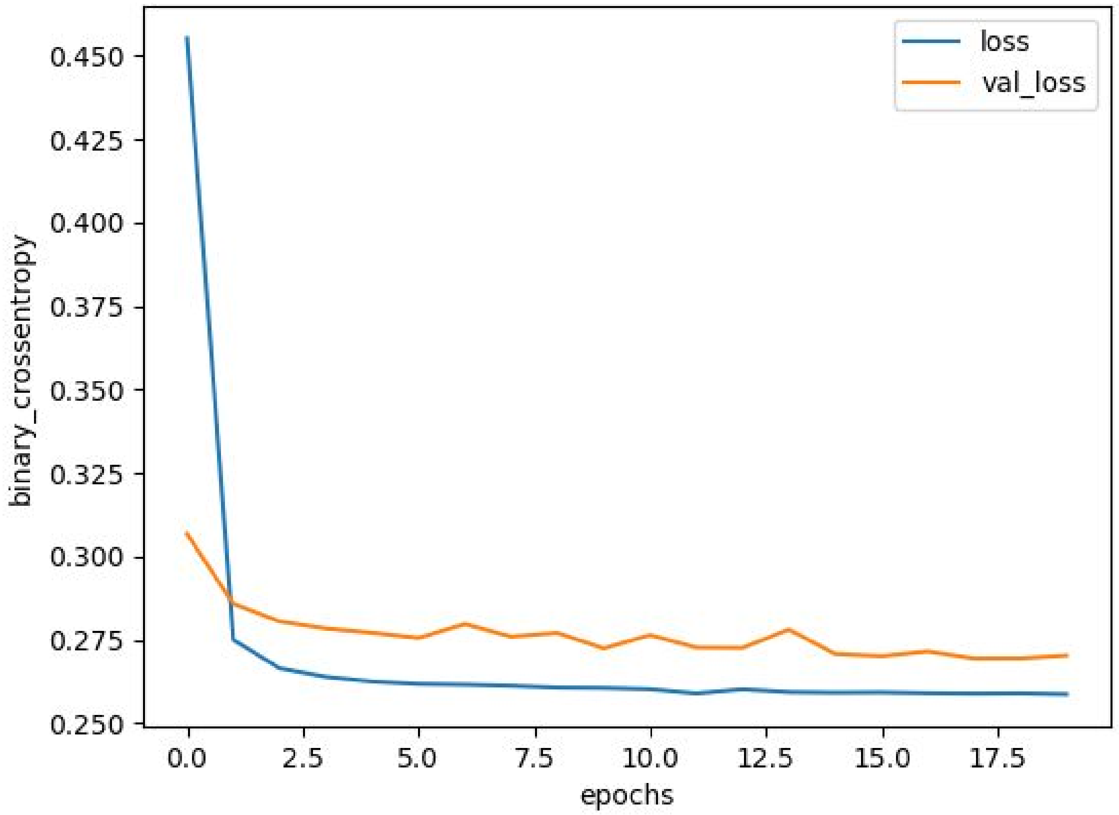
Total number of Parameters : 391

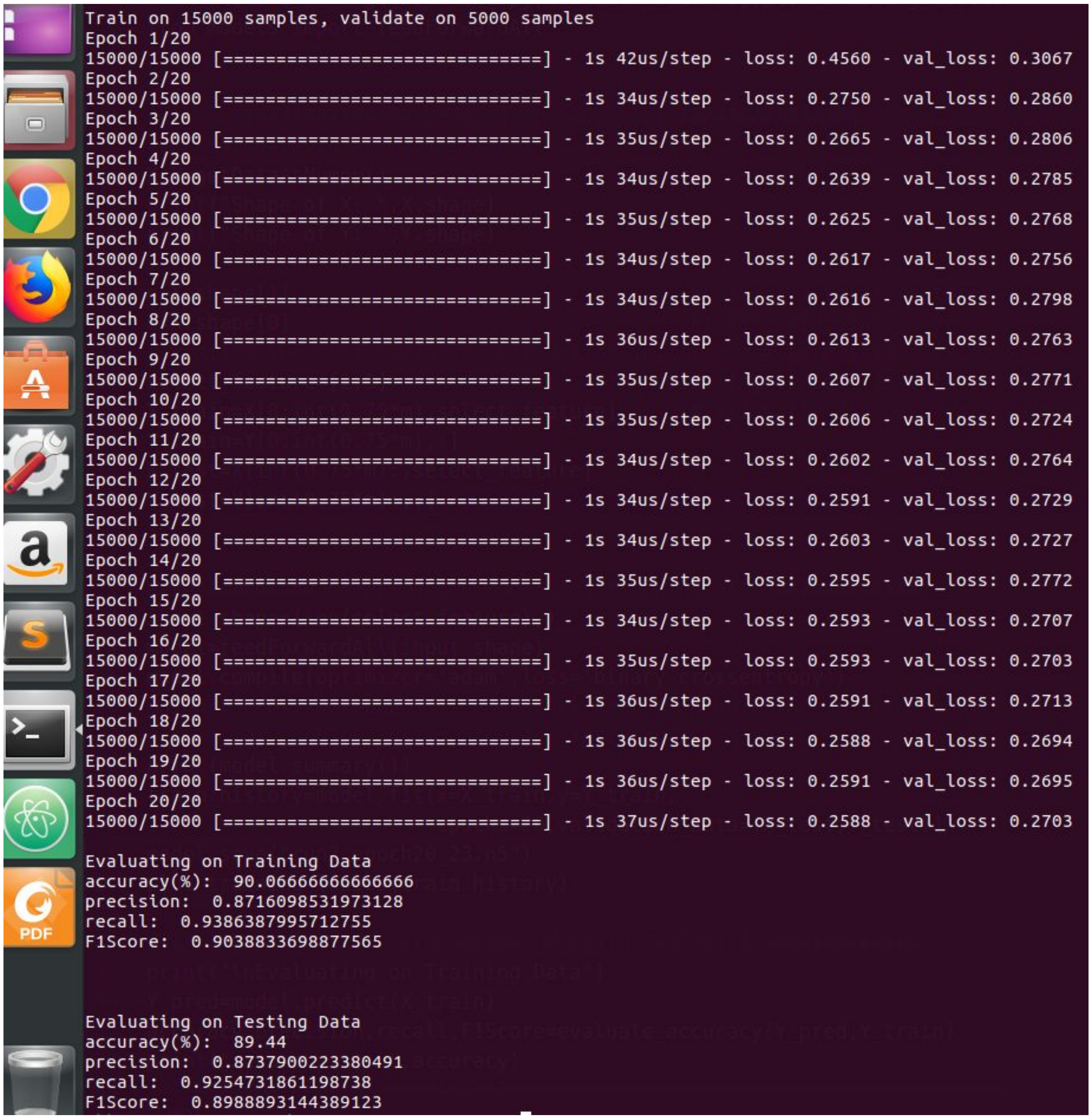
Network Summary (Keras generated) :



```
Shape of X:  (20000, 6)
Shape of Y:  (20000, 1)
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     (None, 2)                 0
_________________________________________________________________
layer1 (Dense)               (None, 20)                60
_________________________________________________________________
layer2 (Dense)               (None, 10)                210
_________________________________________________________________
layer3 (Dense)               (None, 10)                110
_________________________________________________________________
output_layer (Dense)         (None, 1)                 11
=================================================================
Total params: 391
Trainable params: 391
Non-trainable params: 0
```

Training Loss Curve and Accuracy:




```
Train on 15000 samples, validate on 5000 samples
Epoch 1/20
15000/15000 [==============================] - 1s 42us/step - loss: 0.4560 - val_loss: 0.3067
Epoch 2/20
15000/15000 [==============================] - 1s 34us/step - loss: 0.2750 - val_loss: 0.2860
Epoch 3/20
15000/15000 [==============================] - 1s 35us/step - loss: 0.2665 - val_loss: 0.2806
Epoch 4/20
15000/15000 [==============================] - 1s 34us/step - loss: 0.2639 - val_loss: 0.2785
Epoch 5/20
15000/15000 [==============================] - 1s 35us/step - loss: 0.2625 - val_loss: 0.2768
Epoch 6/20
15000/15000 [==============================] - 1s 34us/step - loss: 0.2617 - val_loss: 0.2756
Epoch 7/20
15000/15000 [==============================] - 1s 34us/step - loss: 0.2616 - val_loss: 0.2798
Epoch 8/20
15000/15000 [==============================] - 1s 36us/step - loss: 0.2613 - val_loss: 0.2763
Epoch 9/20
15000/15000 [==============================] - 1s 35us/step - loss: 0.2607 - val_loss: 0.2771
Epoch 10/20
15000/15000 [==============================] - 1s 35us/step - loss: 0.2606 - val_loss: 0.2724
Epoch 11/20
15000/15000 [==============================] - 1s 34us/step - loss: 0.2602 - val_loss: 0.2764
Epoch 12/20
15000/15000 [==============================] - 1s 34us/step - loss: 0.2591 - val_loss: 0.2729
Epoch 13/20
15000/15000 [==============================] - 1s 34us/step - loss: 0.2603 - val_loss: 0.2727
Epoch 14/20
15000/15000 [==============================] - 1s 35us/step - loss: 0.2595 - val_loss: 0.2772
Epoch 15/20
15000/15000 [==============================] - 1s 34us/step - loss: 0.2593 - val_loss: 0.2707
Epoch 16/20
15000/15000 [==============================] - 1s 35us/step - loss: 0.2593 - val_loss: 0.2703
Epoch 17/20
15000/15000 [==============================] - 1s 36us/step - loss: 0.2591 - val_loss: 0.2713
Epoch 18/20
15000/15000 [==============================] - 1s 36us/step - loss: 0.2588 - val_loss: 0.2694
Epoch 19/20
15000/15000 [==============================] - 1s 36us/step - loss: 0.2591 - val_loss: 0.2695
Epoch 20/20
15000/15000 [==============================] - 1s 37us/step - loss: 0.2588 - val_loss: 0.2703

Evaluating on Training Data
accuracy(%):  90.06666666666666
precision:  0.8716098531973128
recall:  0.9386387995712755
F1Score:  0.9038833698877565


Evaluating on Testing Data
accuracy(%):  89.44
precision:  0.8737900223380491
recall:  0.9254731861198738
F1Score:  0.8988893144389123
```

Overall Comparison(on Test Set) on All different Feature Combination

Features Used	Accuracy	Precision	Recall	F1 Score
[0,1,2,3,4,5]	97.82	.96	0.99	0.97
[2,3]	89.42	0.87	0.92	0.89
[0,1]	72.8	0.75	0.69	0.72
[0,2]	82.46	0.80	0.85	0.83
[0,3]	87.06	0.83	0.93	0.87
[0,4]	76.84	0.78	0.74	0.76
[0,5]	73.26	0.73	0.72	0.73
[1,2]	77.72	0.81	0.72	0.76
[1,3]	86.38	0.82	0.92	0.87
[1,4]	59.4	0.58	0.66	0.62
[1,5]	53.94	0.53	0.67	0.59
[2,4]	79.22	0.83	0.73	0.78
[2,5]	77.6	0.81	0.72	0.76
[3,4]	88.98	0.85	0.94	0.89
[3,5]	86.61	0.83	0.92	0.87
[4,5]	59.54	0.59	0.62	0.61

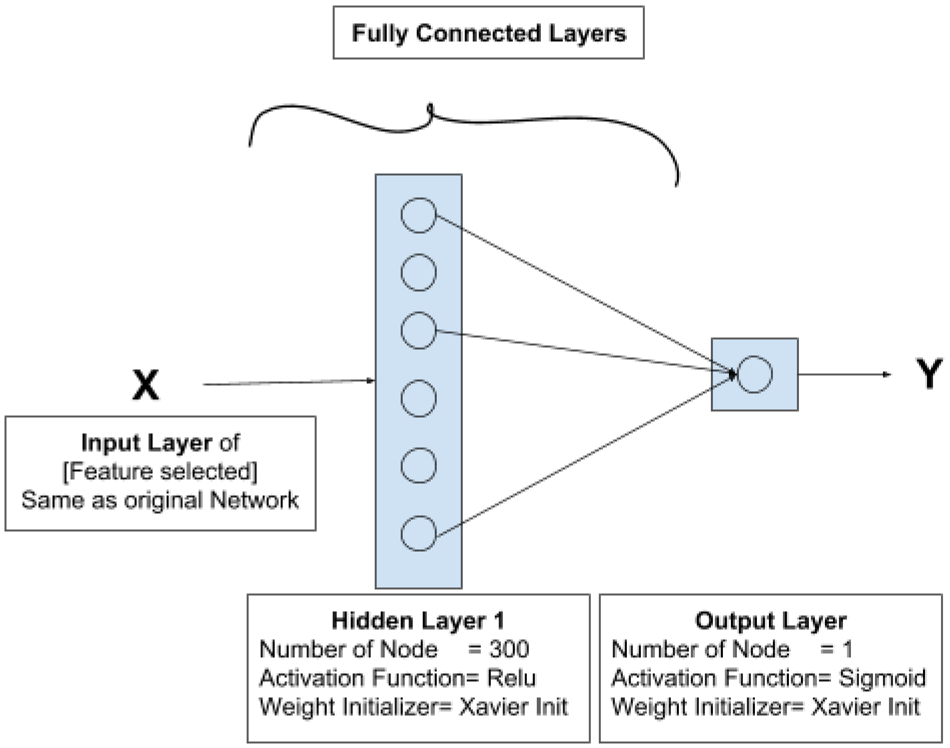
Hence, from above result my Hypothesis is validated that **2nd and 3rd** feature are most important for classifying data and out of them 3rd feature is more important as it is single handedly able to give good accuracy even if it’s paired with bad features.

Apart from that the features **1,4,5** were bad as can be inferred from both intuitive by correlation plot and theoretical support of correlation plot along with the experimental evidence as seen above.

DRONE Neural Network

The drone Neural Network was created in Keras, which take the same input as our Original Neural Network.
The current implementation of Drone Neural Net part in correct but **not very efficient** as each time the extension is triggered I am creating a new Drone Model and initializing the weights from the previous model and the extension part of weight is initialized with zero to ensure continuity of cost function(which I checked manually while testing)
Training Epoch : 20 (kept same as Original Network)

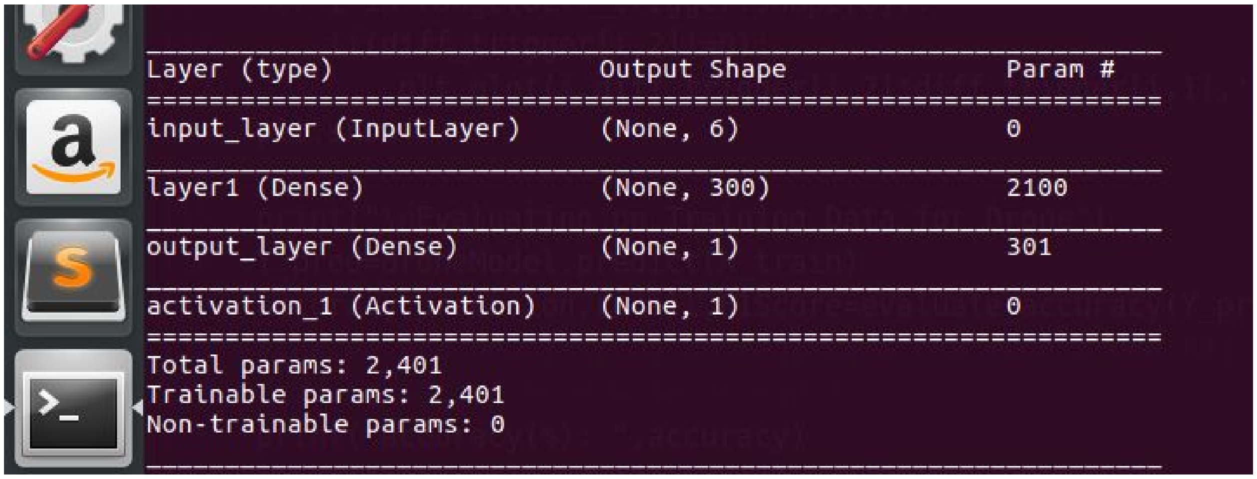
Drone Architecture:



TEST 1: (Teacher-Student combination, ALL 6 Feature)

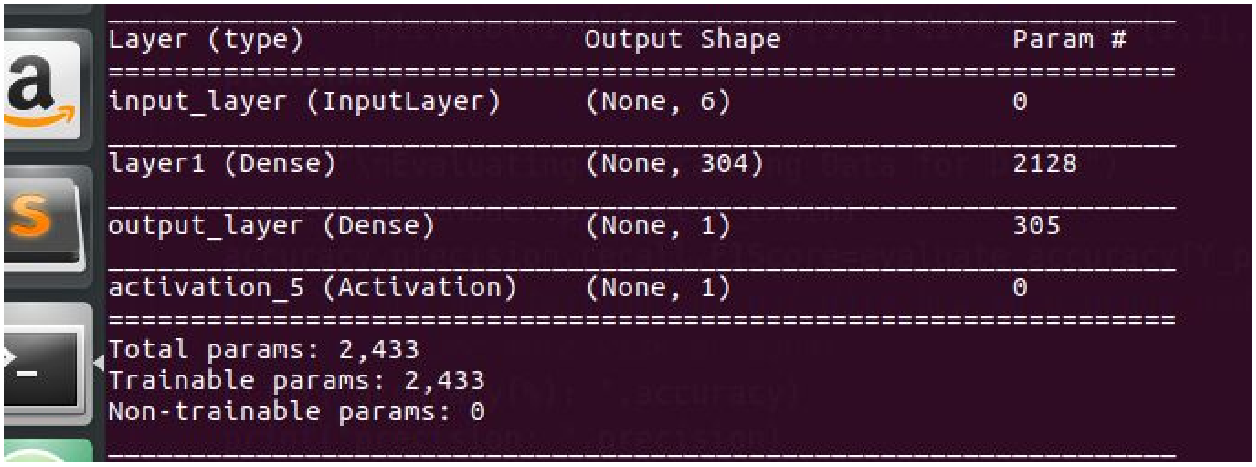
The first test was conducted using all the feature as Test 1 of the original Network. First the Original/Teacher network was trained for 20 epoch. Then the Drone Network was trained using the same input data but its target as the prediction from the Trained Original/Teacher network.
Currently I have just used the **first criteria** for triggering the extension of Drone Network.

Initial Drone Architecture (before training):



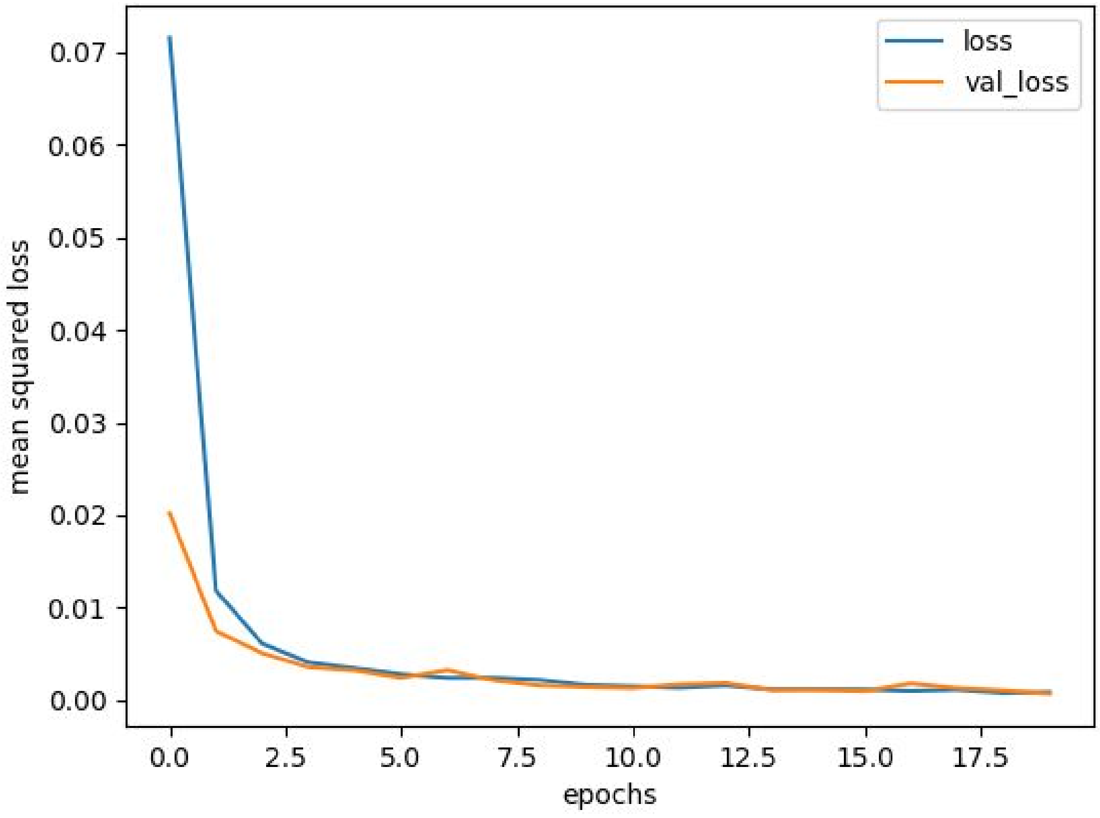
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     (None, 6)                 0
_________________________________________________________________
layer1 (Dense)               (None, 300)               2100
_________________________________________________________________
output_layer (Dense)         (None, 1)                 301
_________________________________________________________________
activation_1 (Activation)    (None, 1)                 0
=================================================================
Total params: 2,401
Trainable params: 2,401
Non-trainable params: 0
_________________________________________________________________
```

Final Drone Architecture (after training)

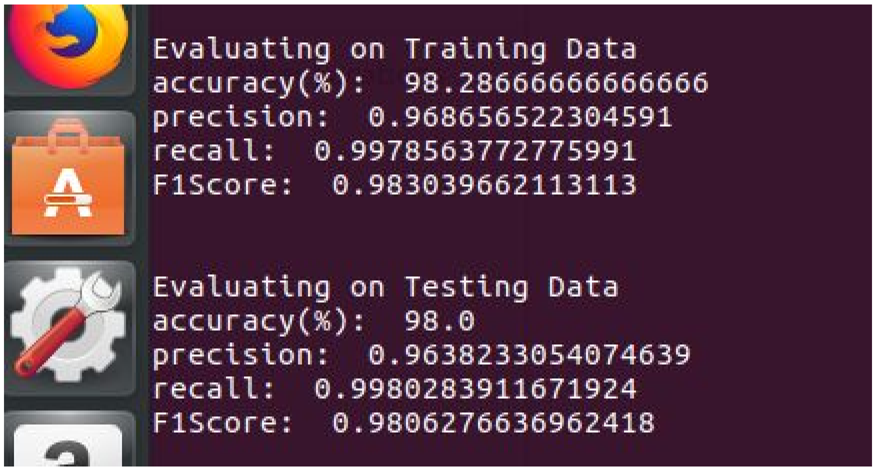


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     (None, 6)                 0
_________________________________________________________________
layer1 (Dense)               (None, 304)               2128
_________________________________________________________________
output_layer (Dense)         (None, 1)                 305
_________________________________________________________________
activation_5 (Activation)    (None, 1)                 0
=================================================================
Total params: 2,433
Trainable params: 2,433
Non-trainable params: 0
_________________________________________________________________
```

Loss Curve for Drone Network:

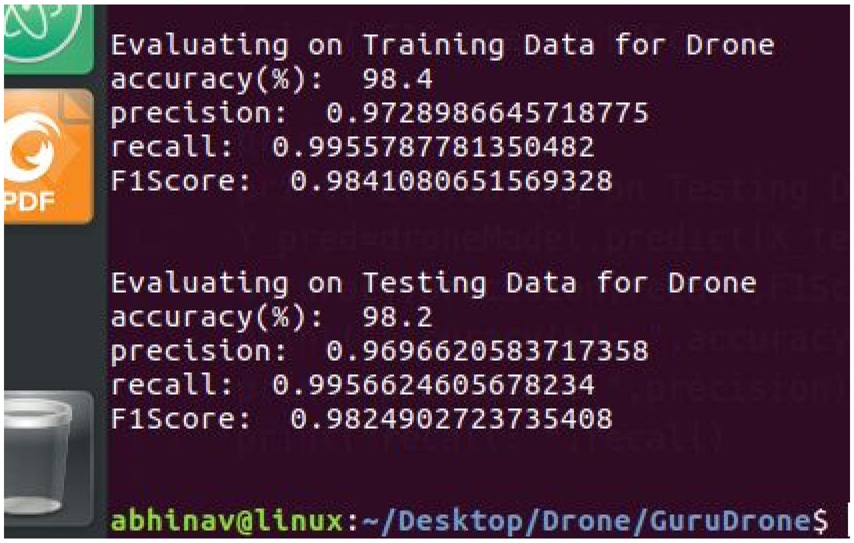


Prediction Accuracy of Original Network/Teacher:



```
Evaluating on Training Data
accuracy(%):  98.28666666666666
precision:  0.968656522304591
recall:  0.9978563772775991
F1Score:  0.983039662113113


Evaluating on Testing Data
accuracy(%):  98.0
precision:  0.9638233054074639
recall:  0.9980283911671924
F1Score:  0.9806276636962418
```

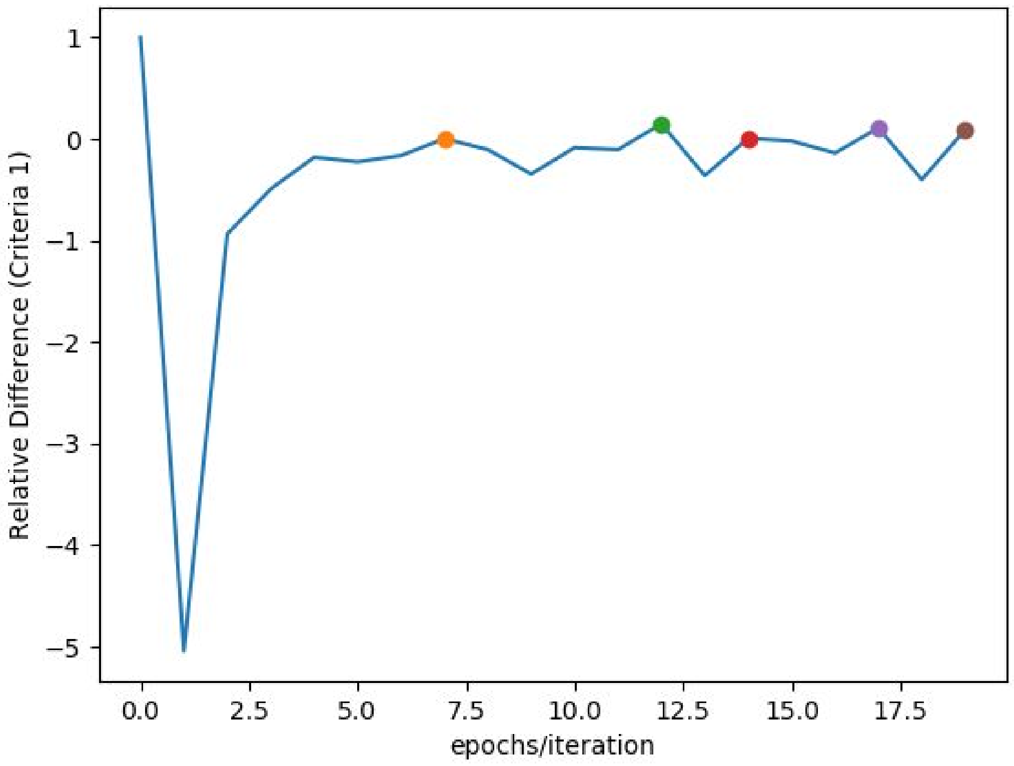
Final Prediction Accuracy of Drone Network:;



```
Evaluating on Training Data for Drone
accuracy(%):  98.4
precision:  0.9728986645718775
recall:  0.9955787781350482
F1Score:  0.9841080651569328

Evaluating on Testing Data for Drone
accuracy(%):  98.2
precision:  0.9696620583717358
recall:  0.9956624605678234
F1Score:  0.9824902723735408

abhinav@linux:~/Desktop/Drone/GuruDrone$
```

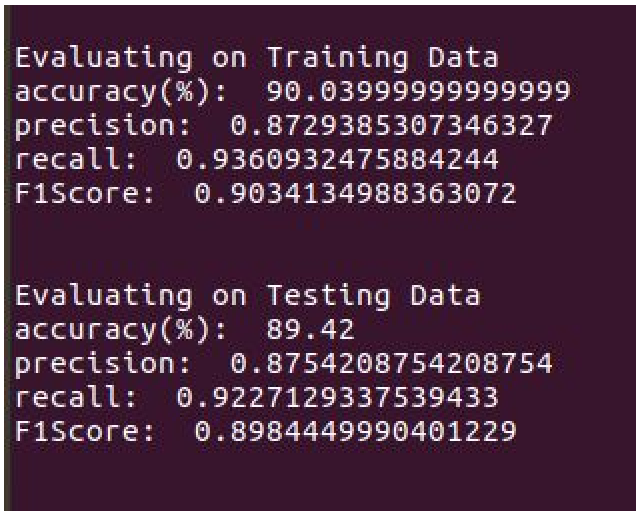
We can see that Drone has also learned the same representation as learned by original Network in 20 epochs. Also, the accuracy on training set is similar to that of testing set hence this shows that the Drone is not overfitting/memorizing the Teacher and like a good student “understanding” the real distinction behind signal and background.

Triggering of Drone Network : (Kappa set to -0.02)
The dots shows the position when the Drone network actually extended its size.



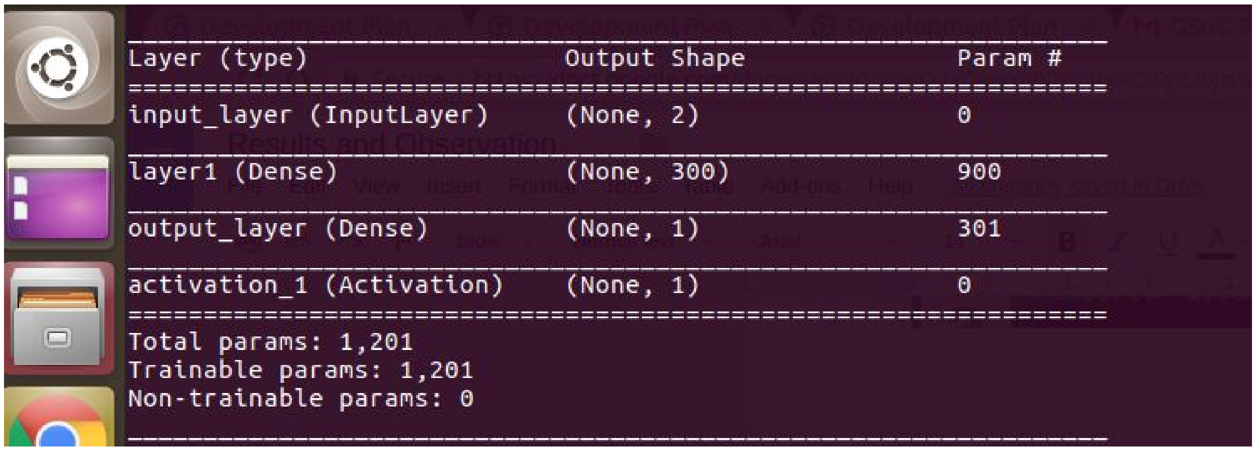
Test on Drone network on Feature 2,3

Original(Teacher) Model Accuracy:



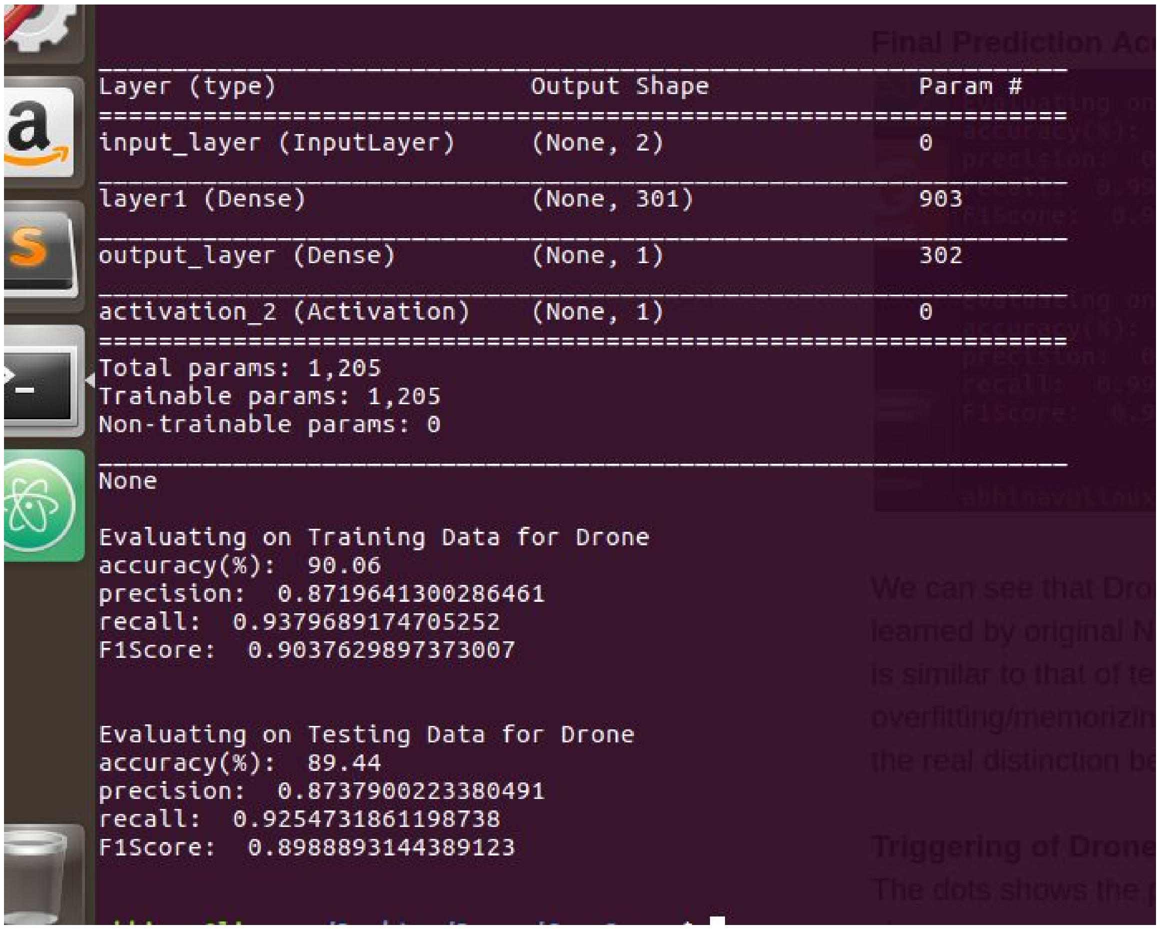
```
Evaluating on Training Data
accuracy(%):  90.03999999999999
precision:  0.8729385307346327
recall:  0.9360932475884244
F1Score:  0.9034134988363072


Evaluating on Testing Data
accuracy(%):  89.42
precision:  0.8754208754208754
recall:  0.9227129337539433
F1Score:  0.8984449990401229
```

Initial Drone Architecture (before training):



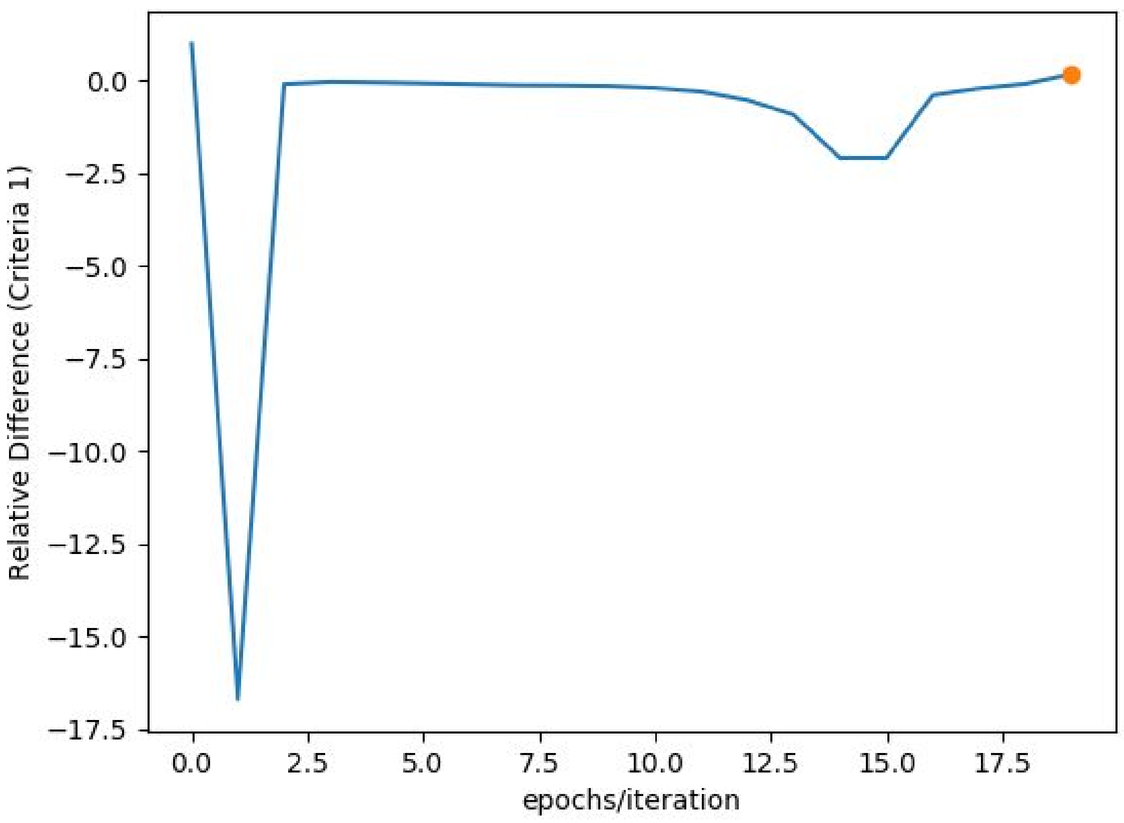
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     (None, 2)                 0
_________________________________________________________________
layer1 (Dense)               (None, 300)               900
_________________________________________________________________
output_layer (Dense)         (None, 1)                 301
_________________________________________________________________
activation_1 (Activation)    (None, 1)                 0
=================================================================
Total params: 1,201
Trainable params: 1,201
Non-trainable params: 0
_________________________________________________________________
```

Final Drone Architecture (after training) and accuracy of Drone finally:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_layer (InputLayer)     (None, 2)                 0
_________________________________________________________________
layer1 (Dense)               (None, 301)               903
_________________________________________________________________
output_layer (Dense)         (None, 1)                 302
_________________________________________________________________
activation_2 (Activation)    (None, 1)                 0
=================================================================
Total params: 1,205
Trainable params: 1,205
Non-trainable params: 0
_________________________________________________________________
None

Evaluating on Training Data for Drone
accuracy(%):  90.06
precision:  0.8719641300286461
recall:  0.9379689174705252
F1Score:  0.9037629897373007


Evaluating on Testing Data for Drone
accuracy(%):  89.44
precision:  0.8737900223380491
recall:  0.9254731861198738
F1Score:  0.8988893144389123
```

Drone Extension Plot:


Relative Difference (Criteria 1)
epochs/iteration