# HW 2 MLP ie7860

by Igor Ostaptchenko AKA igor\_ost@wayne.edu

The MLP assignment explores two datasets, one is highly unbalanced and another rather short. First the performance of manually designed networks is evaluated then the hyper-parameter turning is performed using Scikit-Optimize and results compared.

## Setup jupiter in virtial environment

python3 -m venv ds  
. ds/bin/activate  
pip install --upgrade pip  
pip install -r req.txt   
jupyter nbextension enable --py --sys-prefix ipympl  
jupyter notebook &

## Datasets

### Thyroid

* http://archive.ics.uci.edu/ml/datasets/Thyroid+Disease
* http://fizyka.umk.pl/kis-old/projects/datasets.html#Hypothyroid

Thyroid dataset has 21 features (15 binary, 6 continuous) and divided into 3 classes. There are 3772 training and 3428 testing examples; primary hypothyroid, compensated hypothyroid, normal (not hypothyroid).

Training set <ann-train.data>: - 93 of class 1 or 2.47% - 191 of class 2 or 5.06% - 3488 of class 3 or 92.47%

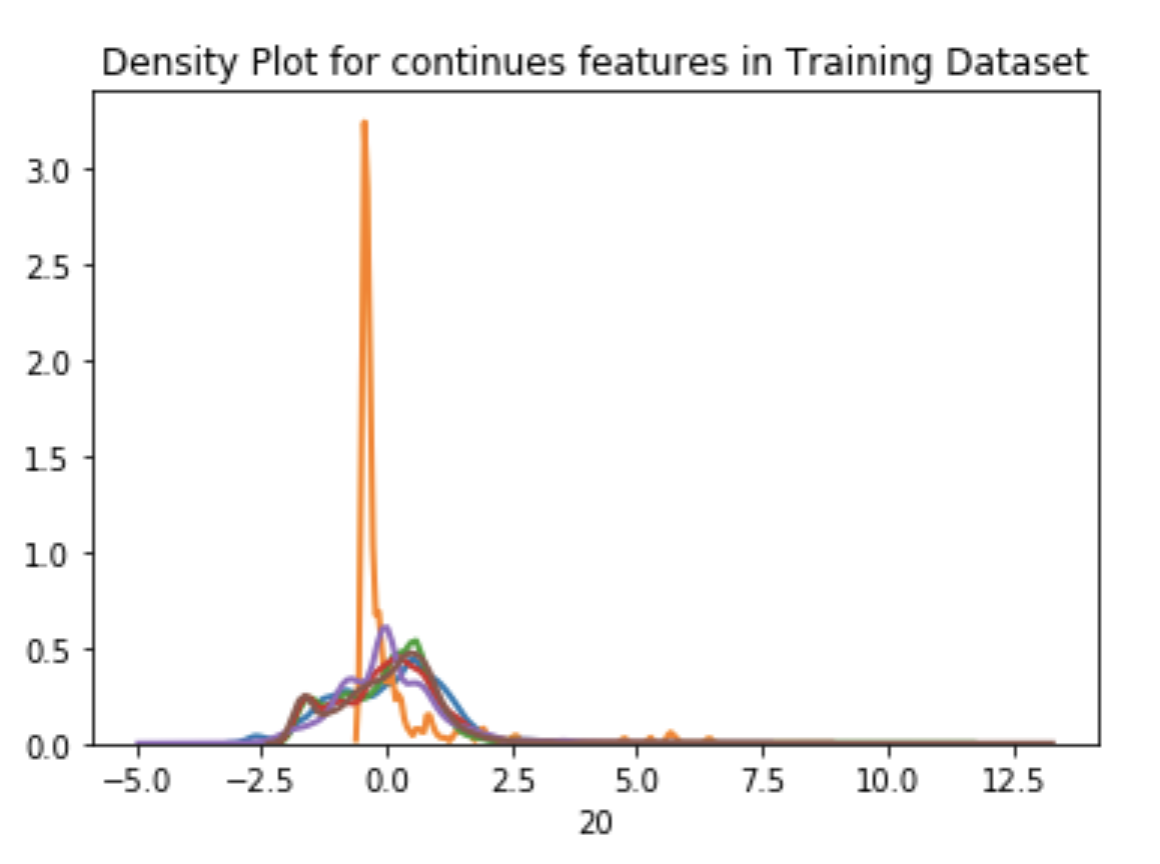
Test set <ann-test.data>: - 73 of class 1 or 2.13% - 177 of class 2 or 5.16% - 3178 of 92.71%

Because 92 percent of the patients are not hyperthyroid the good classifier must be better than 92%. The dataset has imbalanced representation of the classes that poses a problem for trainig Neural Networks. While there is a way to use [Class Wheight](https://www.tensorflow.org/tutorials/structured_data/imbalanced_data) technique, this paper uses [Oversampling of minority class Resampling Technique](https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html).

#### Oversampling of class 1 and 2 of Thyroid Dataset

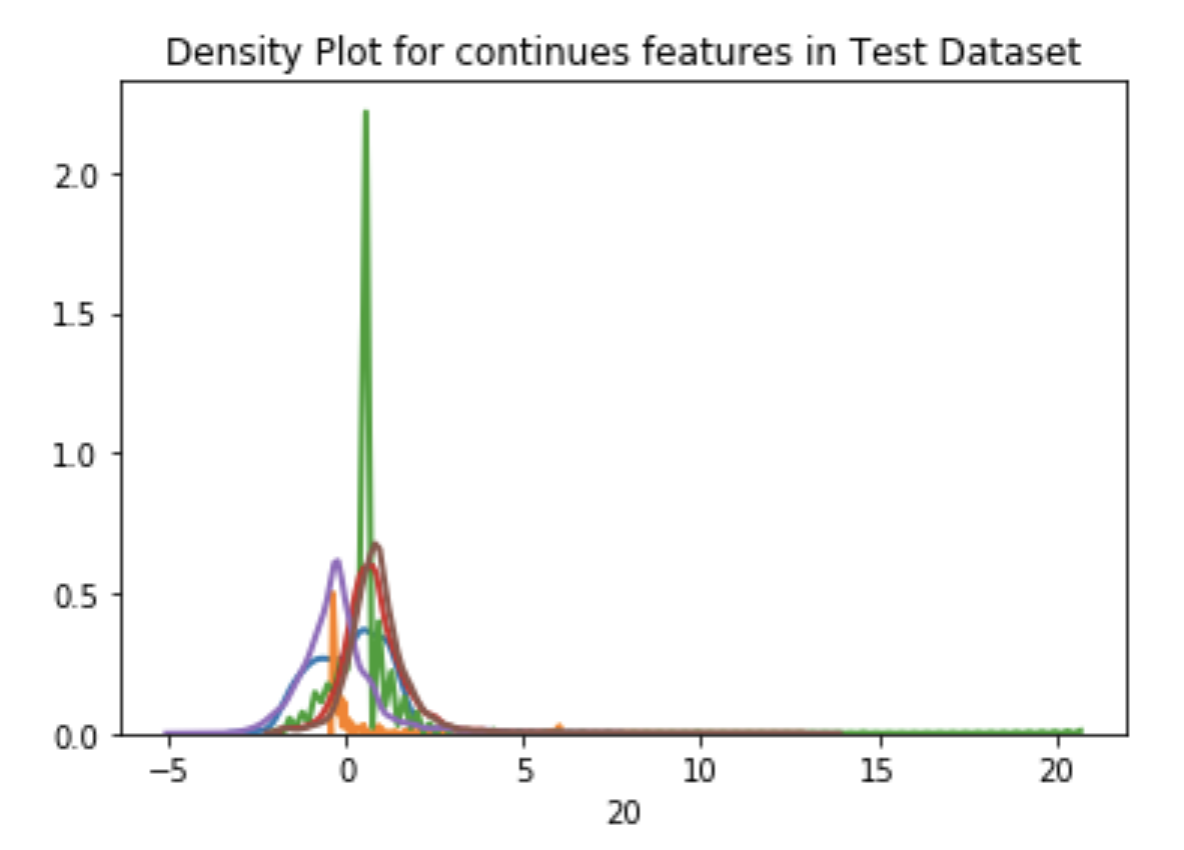
TODO proper Oversampling

#### Density Plot of Continues features of the Training Dataset



thy-train-pdp.png

#### Density Plot of Continues features of the Testing Dataset



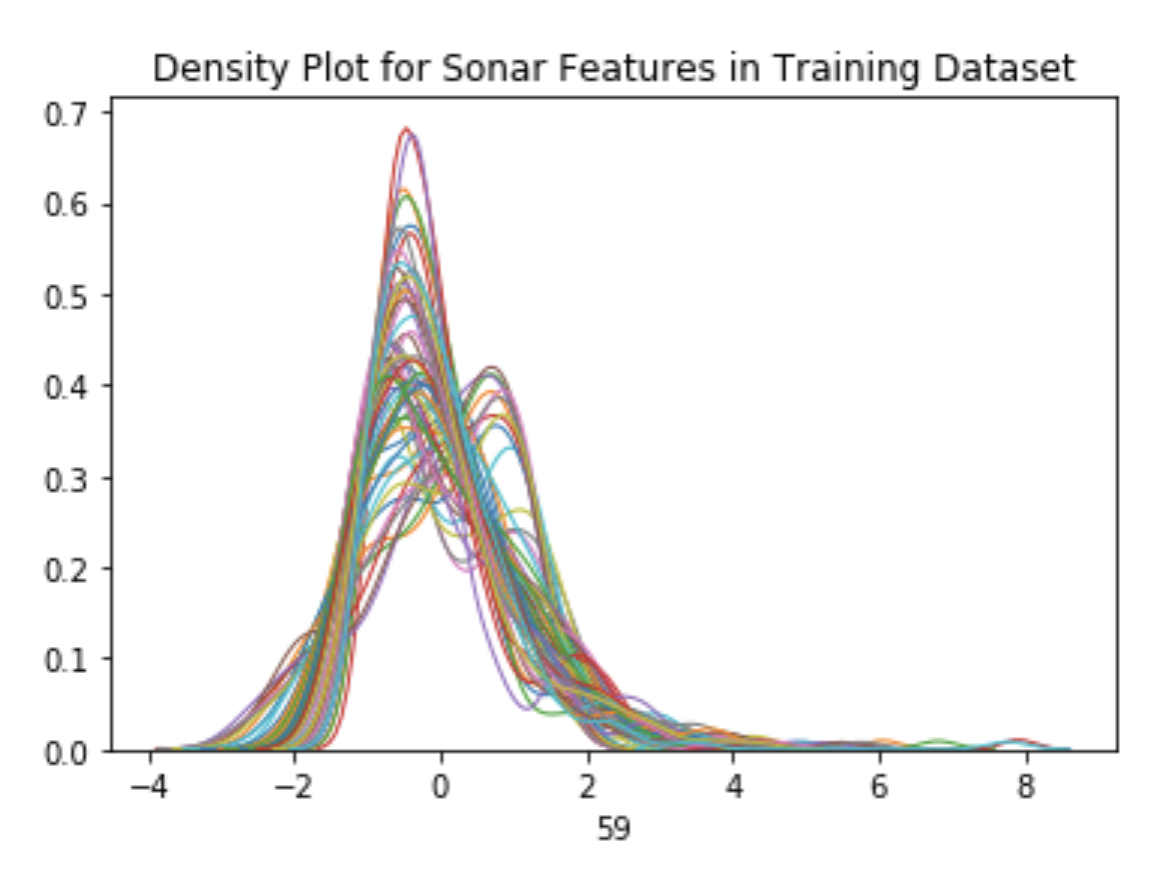
thy-test-pdp.png

### Sonar

* http://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar%2C+Mines+vs.+Rocks) more info
* http://fizyka.umk.pl/kis-old/projects/datasets.html#Sonar

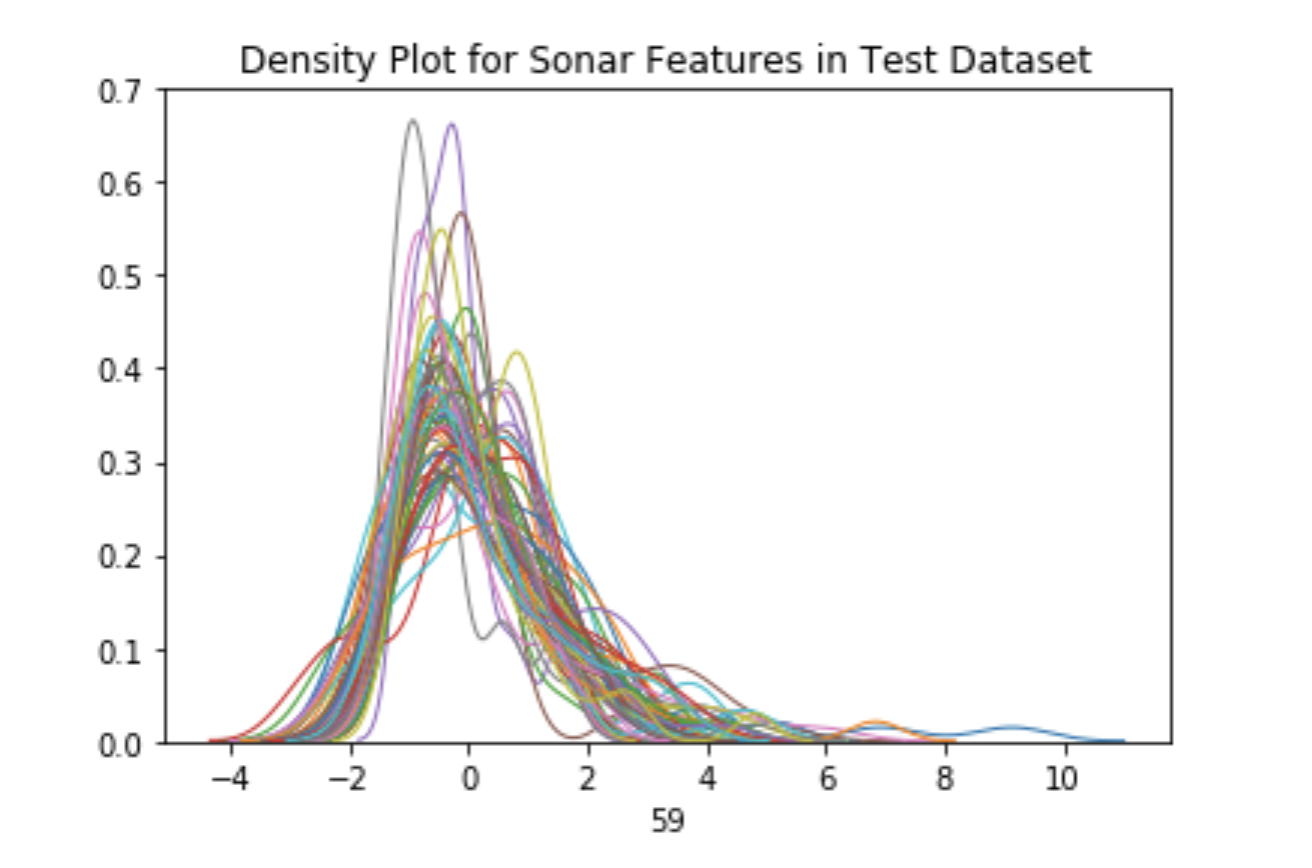
The sonar dataset <sonar.csv> contains data about patterns obtained by bouncing sonar signals off of a metal cylinder at various angles and under various conditions and patterns obtained from rocks under similar conditions. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency. The data set contains signals obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock. Each record has 60 features in the range 0.0 to 1.0.

#### Density Plot of Training Dataset



sonar-train-pdp.png

#### Density Plot of Testing Dataset



sonar-test-pdp.png

In the original dataset the label associated with each record contains the letter “R” if the object is a rock and “M” if it is a mine (metal cylinder). The transformed <sonar.csv> has 1 for Mine and 0 for Rock.

We’ll explore the training of a neural network to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock.

More info about the dataset:

The papers using the dataset: \* [Analysis of hidden units in a layered network trained to classify sonar targets R Paul Gorman Terrence J Sejnowski  
Neural Networks vol 1, pages 75](http://rexa.info/paper/7257d06678a052c7cb6f1d08d8eda2f5ac07f74a)

## Manual Design of NN Model

The number of features of the dataset may give some ideas on the possible network topology. We’ll design a few topologies for each dataset and investigate performance. The next chapter will use the hyper-parameter turning.

### Sonar NN design

The code is located here: <https://github.com/borodark/ie7860/blob/master/HW2%20MLP%20Sonar.ipynb>

The sonar data has 60 input features.

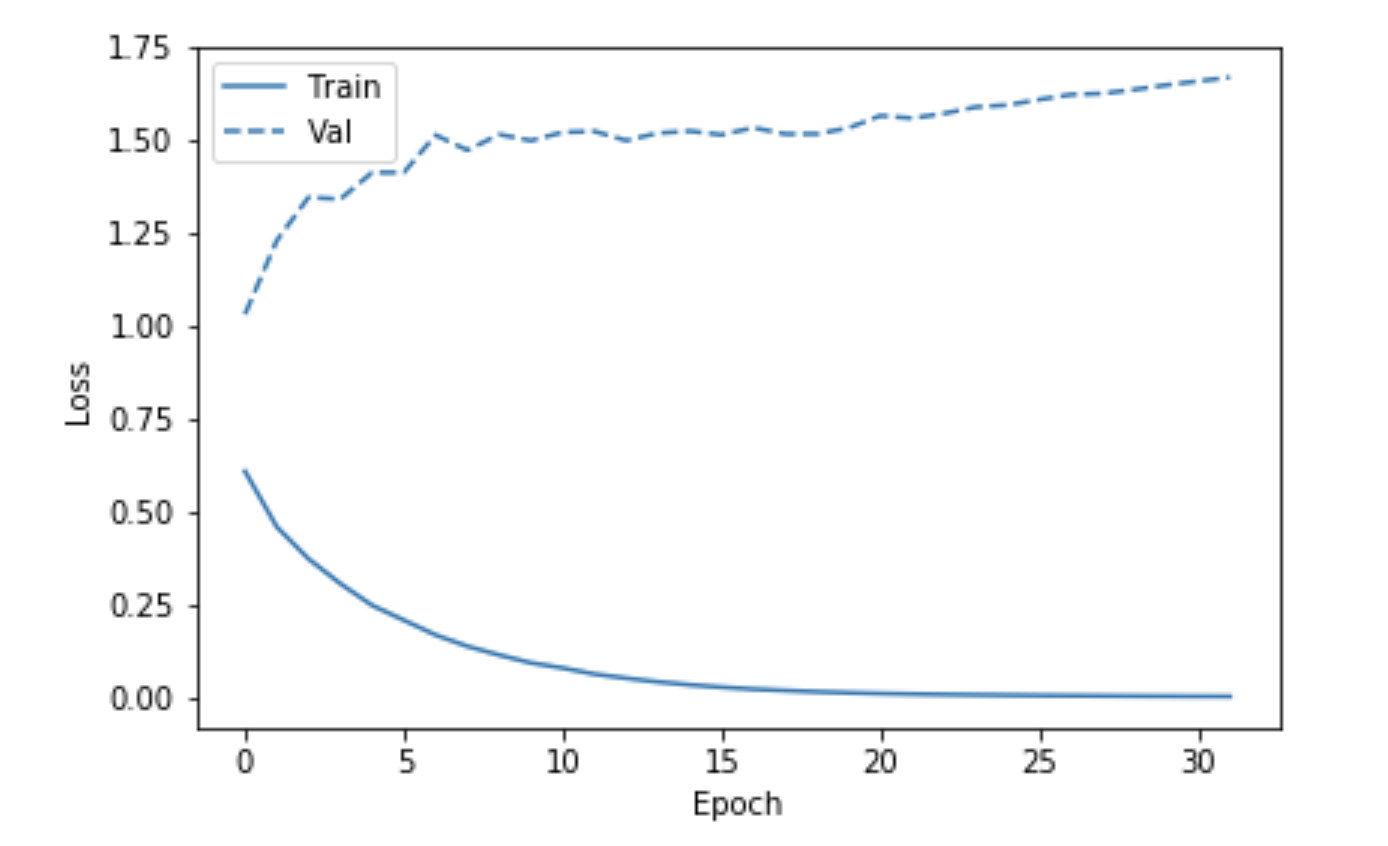
#### The first design is 60-64-16-1

* 64 neurons in the first hidden layer
* 16 neurons in the second hidden layer

##### The process of the fitting

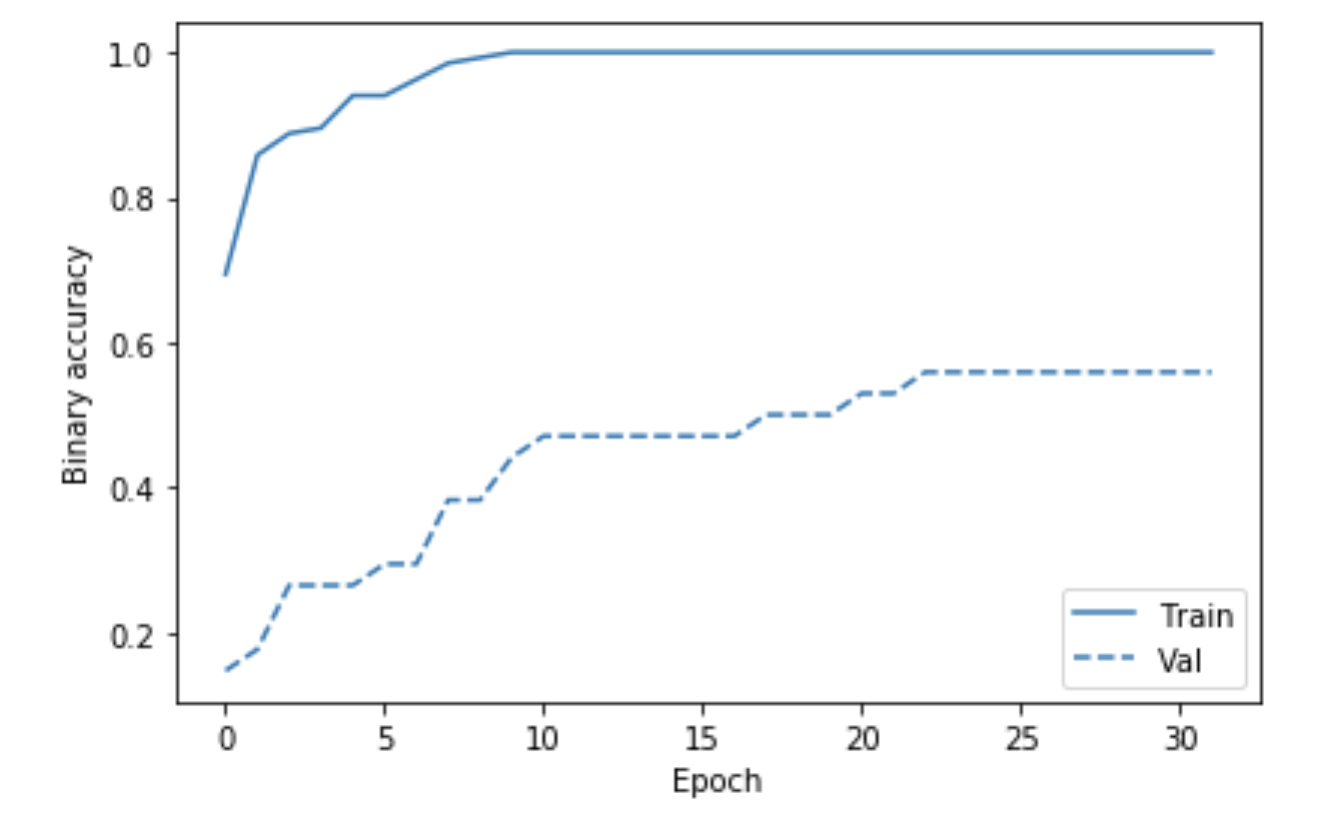
# Defining initialization parameters for 61-64-16-1 MLP model  
num\_classes = y\_train\_one\_hot.shape[1]; num\_features = X\_train.shape[1]  
num\_layers\_0 = 64; num\_layers\_1 = 16  
  
# Define the keras model  
model = Sequential()  
model.add(Dense(num\_layers\_0, input\_dim=num\_features, activation='relu'))  
model.add(Dense(num\_layers\_1, activation='relu'))  
model.add(Dense(num\_classes, activation='sigmoid'))  
  
# Compile the keras model  
model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['binary\_accuracy'])  
# Fit keras model  
history = model.fit(X\_train, y\_train\_one\_hot, epochs=32, batch\_size=4, validation\_split = 0.20)

The Losses during the process



sonar-man-loss1.png

The Binary Accuracy during the process



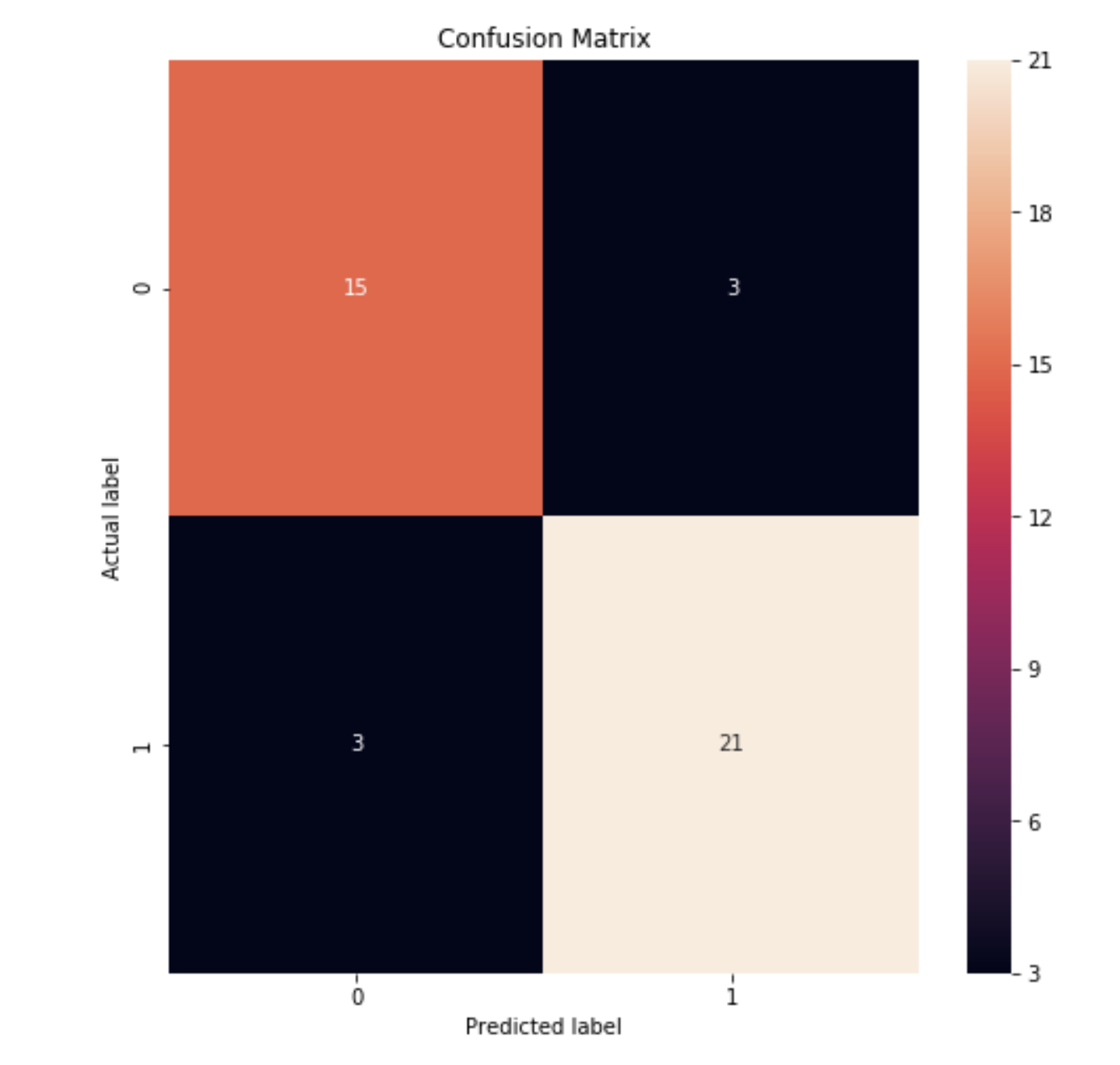
sonar-man-bin-acc1.png

Predictions are made on the test data

y\_test\_predictions = model.predict\_classes(X\_test, batch\_size=1)  
baseline\_results = model.evaluate(X\_test, y\_test\_one\_hot,  
 batch\_size=1, verbose=1)  
  
for name, value in zip(model.metrics\_names, baseline\_results):  
 print(name, ': ', value)  
 print()  
  
plot\_cm(y\_test\_one\_hot, y\_test\_predictions)

The results and confusion matrix is as follows.

loss : 0.58803293892289  
  
binary\_accuracy : 0.8571428571428571



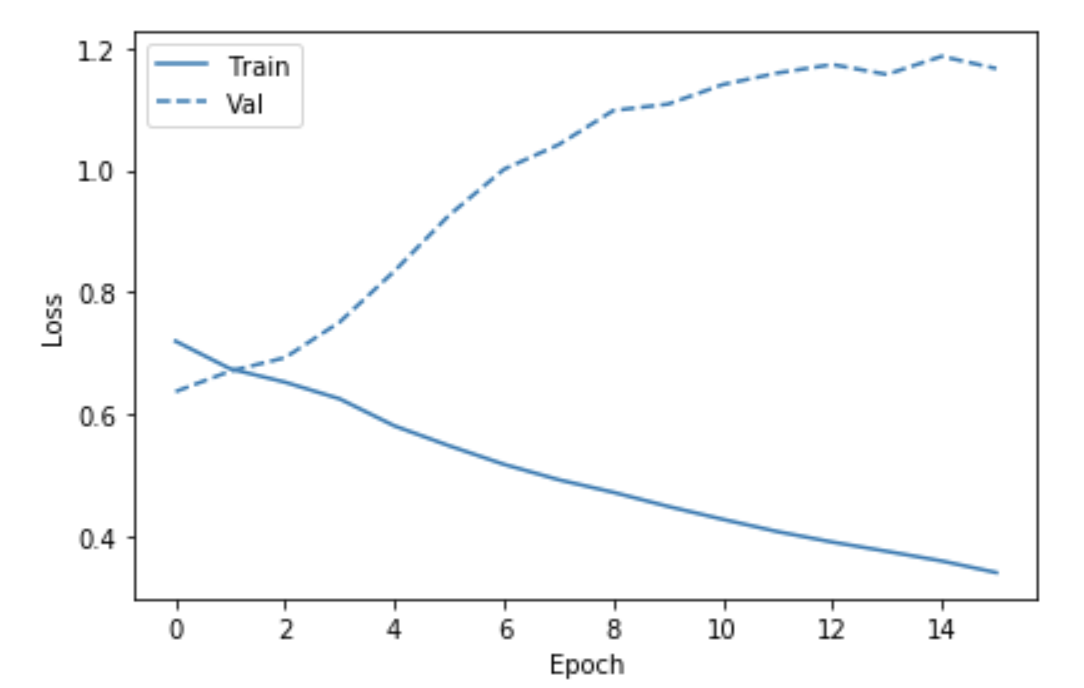
sonar-man-cf-1.png

#### The next is the 3 layers network: 60-64-16-8-1

* 64 neurons in the first hidden layer
* 16 neurons in the second hidden layer
* 8 neurons in the third hidden layer

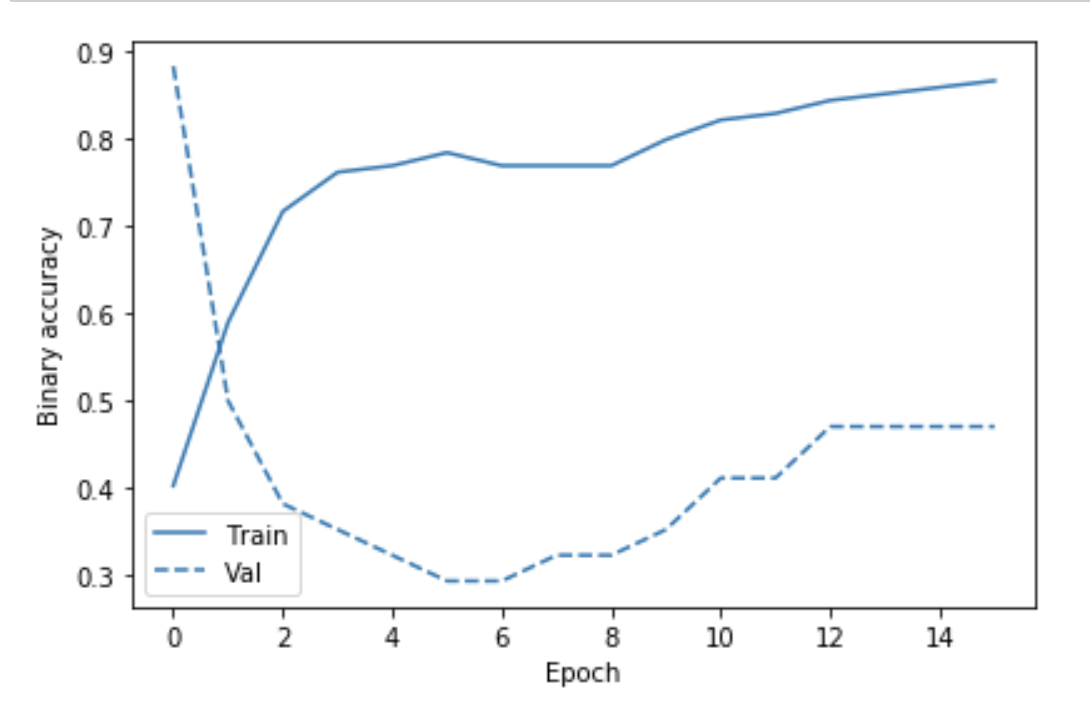
##### The process of the fitting

The Losses during the process



sonar-man-loss2.png

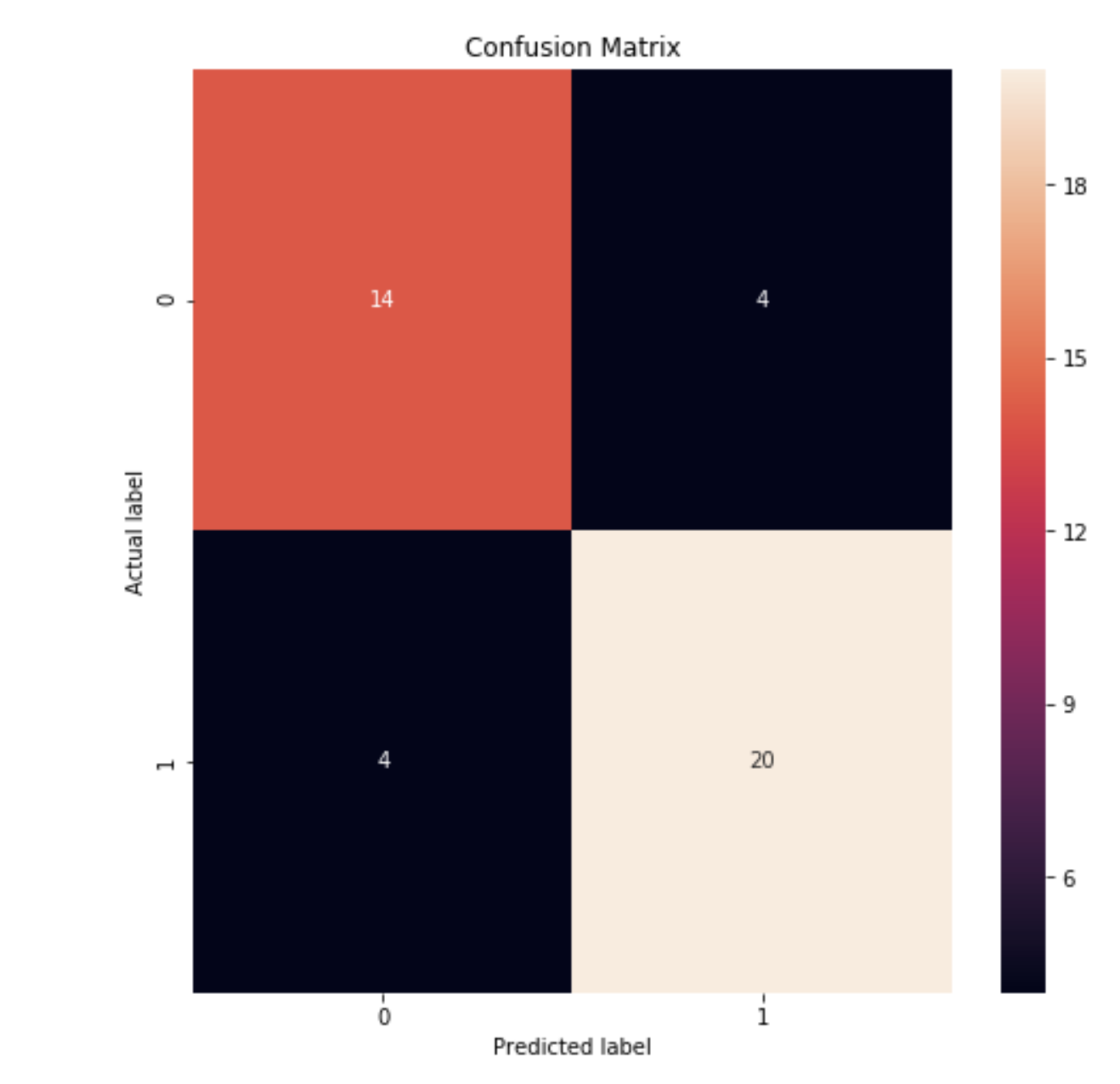
The Binary Accuracy during the process



sonar-man-bin-acc2.png

The results and confusion matrix is as follows.

loss : 0.4862921335512683  
  
binary\_accuracy : 0.8095238095238095



sonar-man-cf-2.png

#### The conclusion

The 2 layer 64-16 network performs slightly better.

### Thyroid NN design

The code is located here: <https://github.com/borodark/ie7860/blob/master/HW2%20MLP%20Thyroid.ipynb>

The Thyroid data has 21 input features.

#### The first design is 21-32-32-1

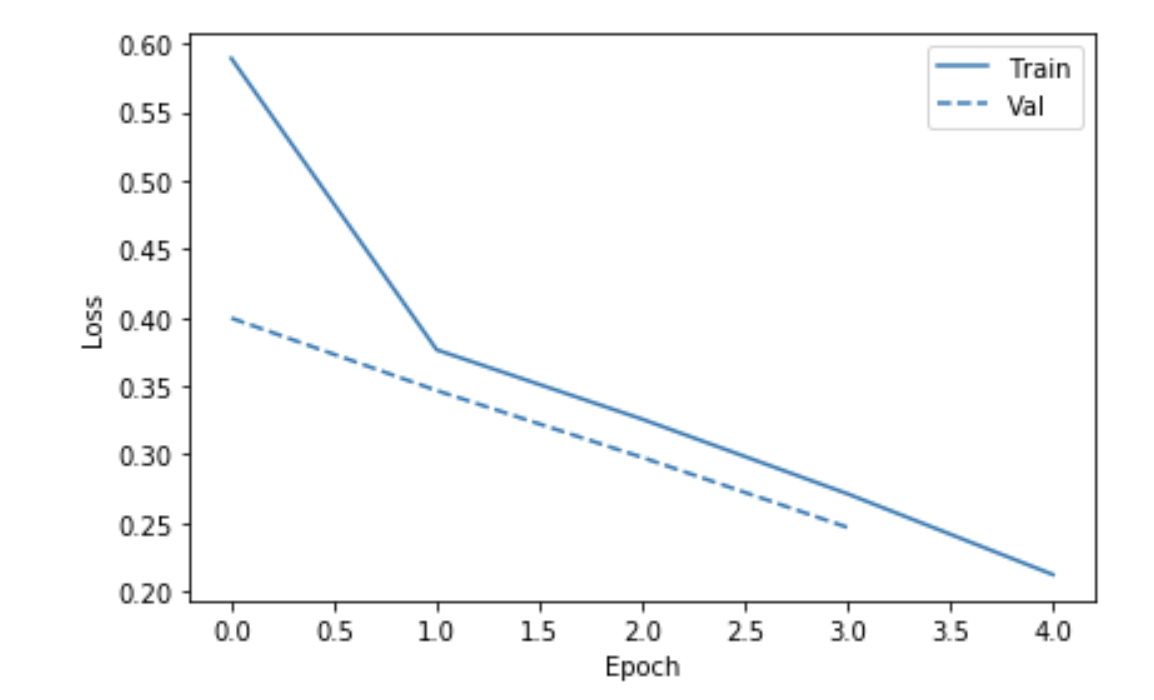
* 32 neurons in the first hidden layer
* 32 neurons in the second hidden layer

The design is not stable: the network collapses if more than 6 epocs is given.

##### The process of the fitting

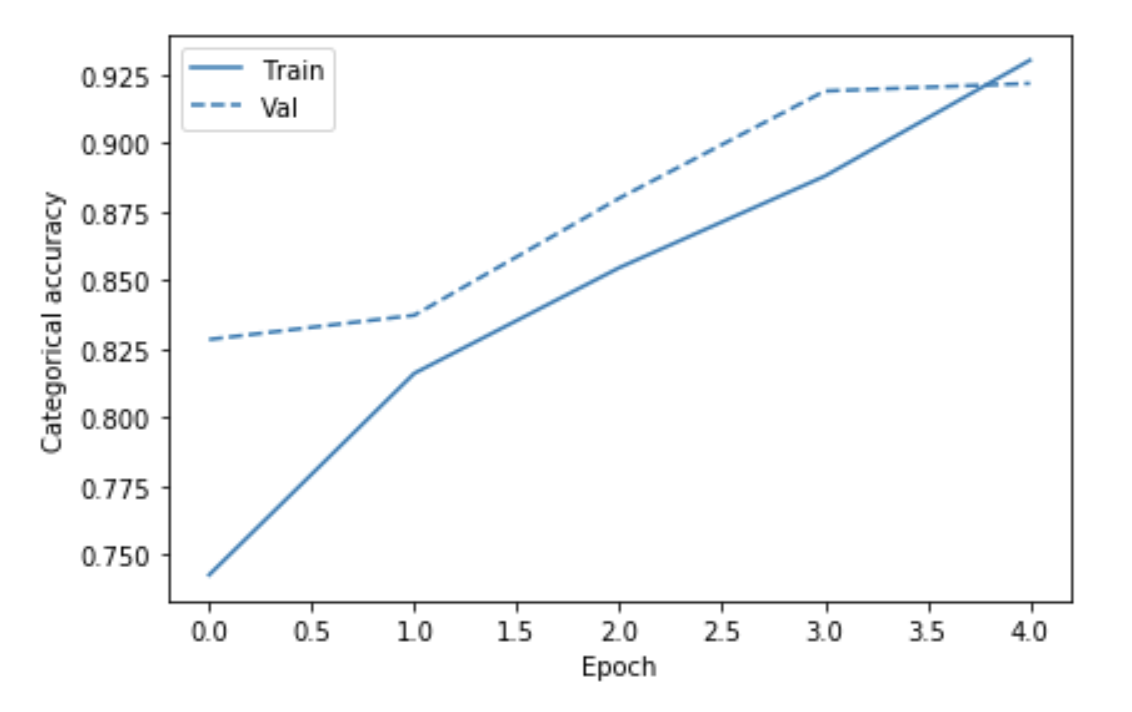
The performance reached on during 5 epocs is not spectacular!

The Losses during the process



thy-man-loss1.png

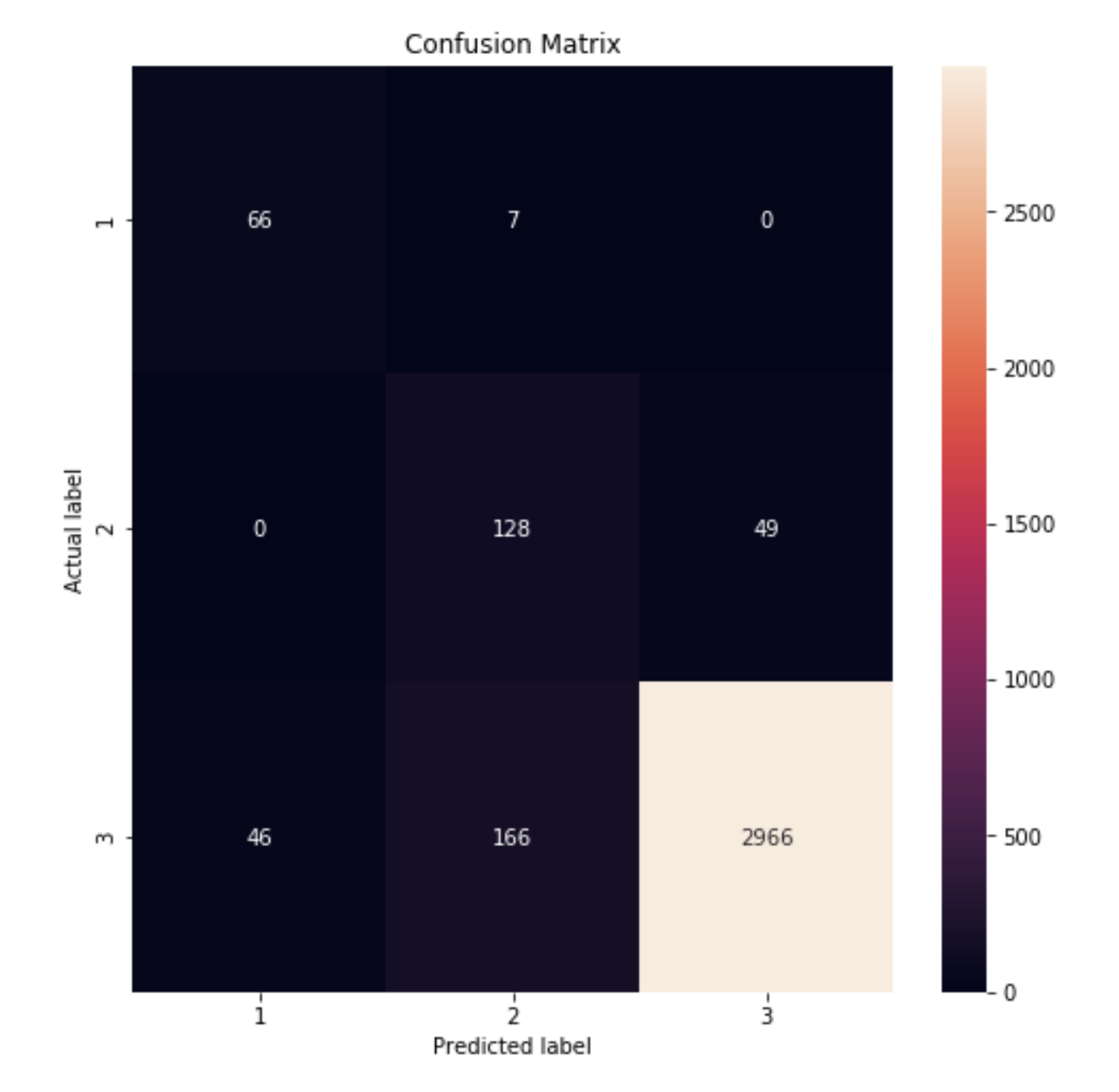
The Binary Accuracy during the process



thy-man-bin-acc1.png

The results and confusion matrix is as follows.

loss : 0.2882766517508153  
  
categorical\_accuracy : 0.9218203033838973



thy-man-cf-1.png

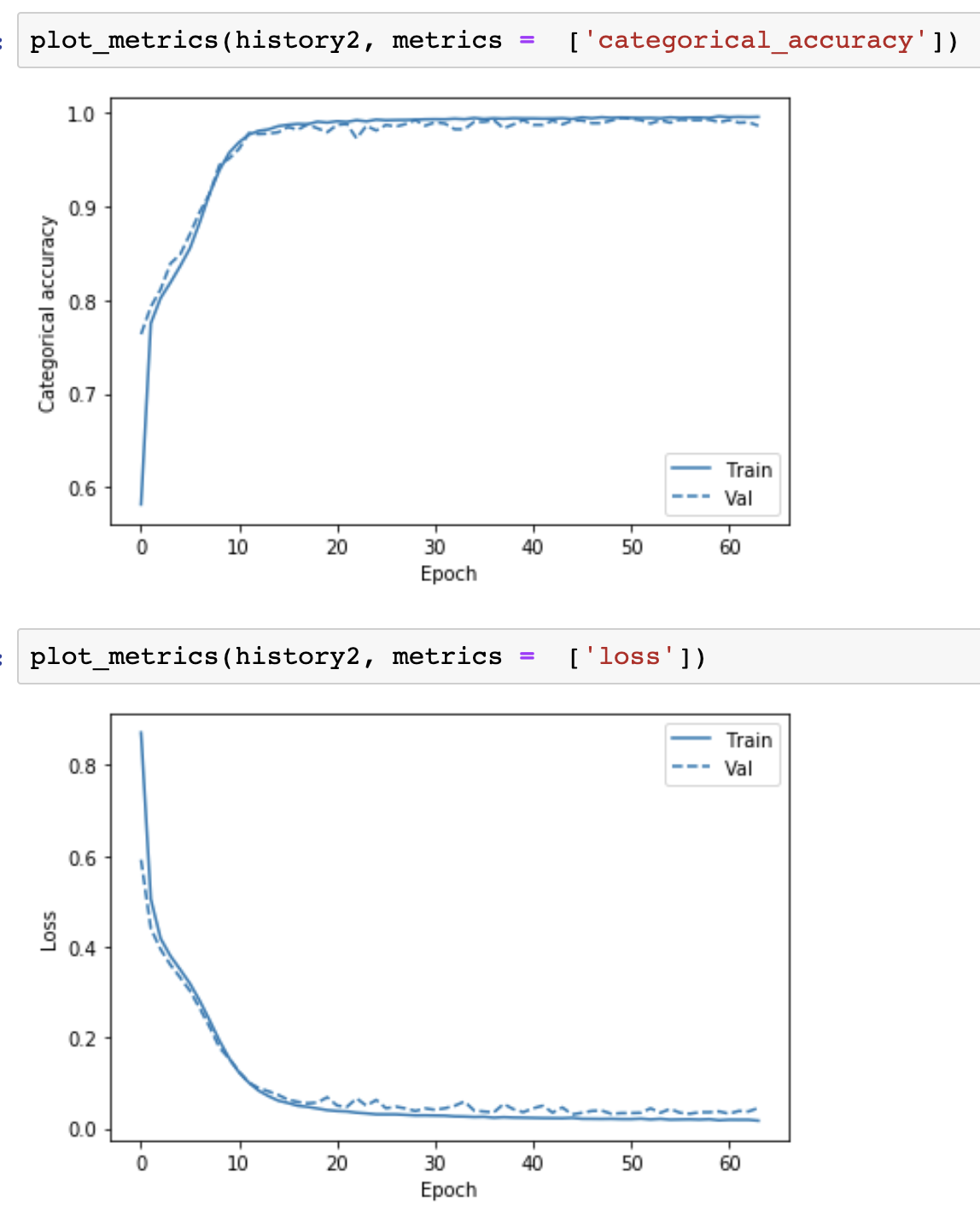
#### The second desing is 21-16-32-16-1

* 16 neurons in the first hidden layer
* 32 neurons in the second hidden layer
* 16 neurons in the third hidden layer

Another change is use of rmsprop optimizer instead of adam.

# Define the keras model 21-16-32-16-1 MLP model  
model2 = Sequential()  
model2.add(Dense(16, input\_dim=21, activation='relu'))  
model2.add(Dense(32, activation='relu'))  
model2.add(Dense(16, activation='sigmoid'))  
model2.add(Dense(3, activation='sigmoid'))  
  
# Compile the keras model  
model2.compile(loss='categorical\_crossentropy', optimizer='rmsprop', metrics=['categorical\_accuracy'])  
# Fit keras model  
history2 = model2.fit(train, y\_train\_one\_hot, epochs=64, validation\_split = 0.20)

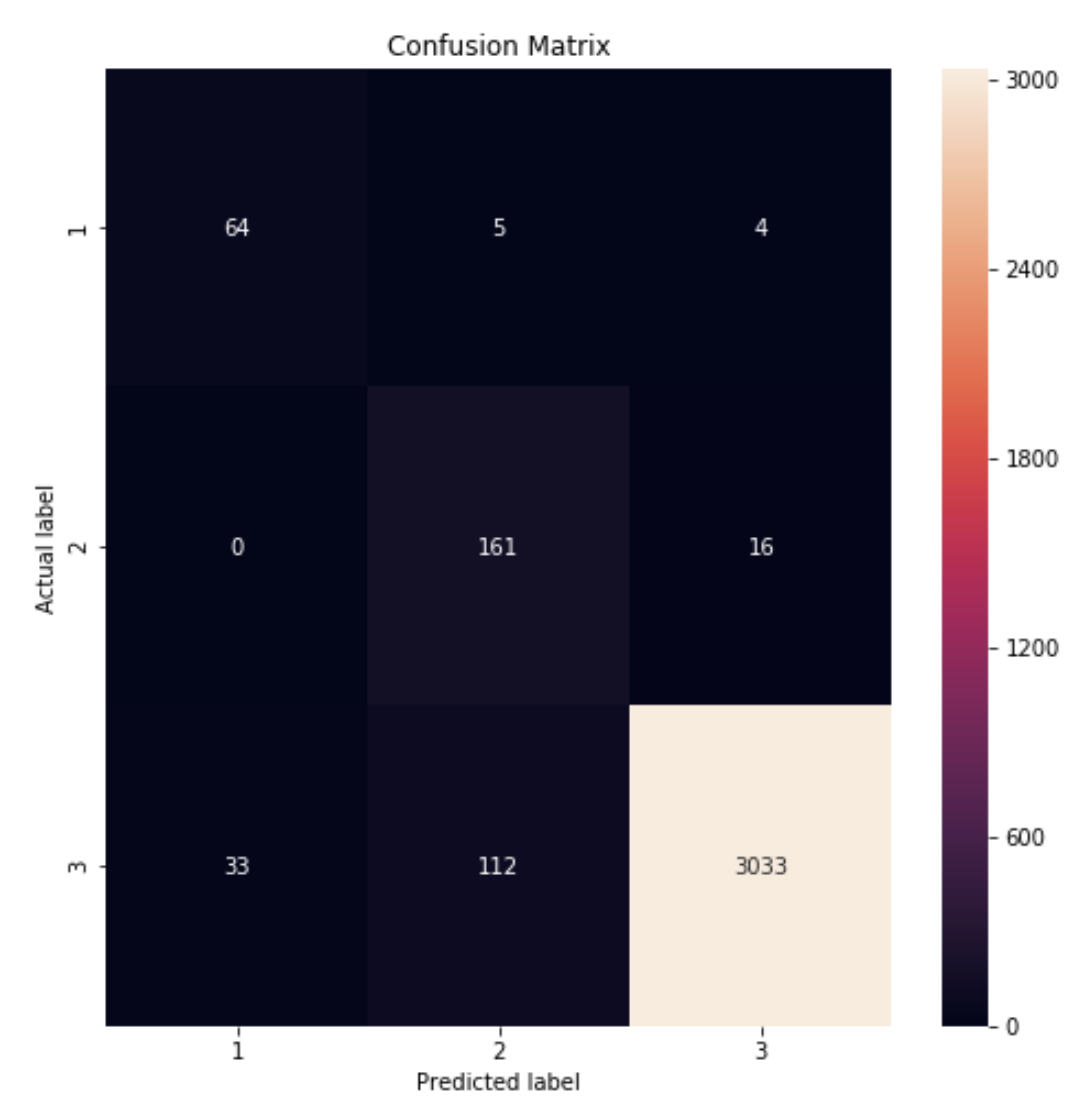
The accuracy and losses plots:



thy-loss-acc-2.png

The results and confusion matrix:

loss : 0.22274850618860179  
  
categorical\_accuracy : 0.9504084014002334



thy-cf-2.png

#### The conclusion

The 3 layer 16-32-16 network performs much better and does not collapse on 64 epocs.

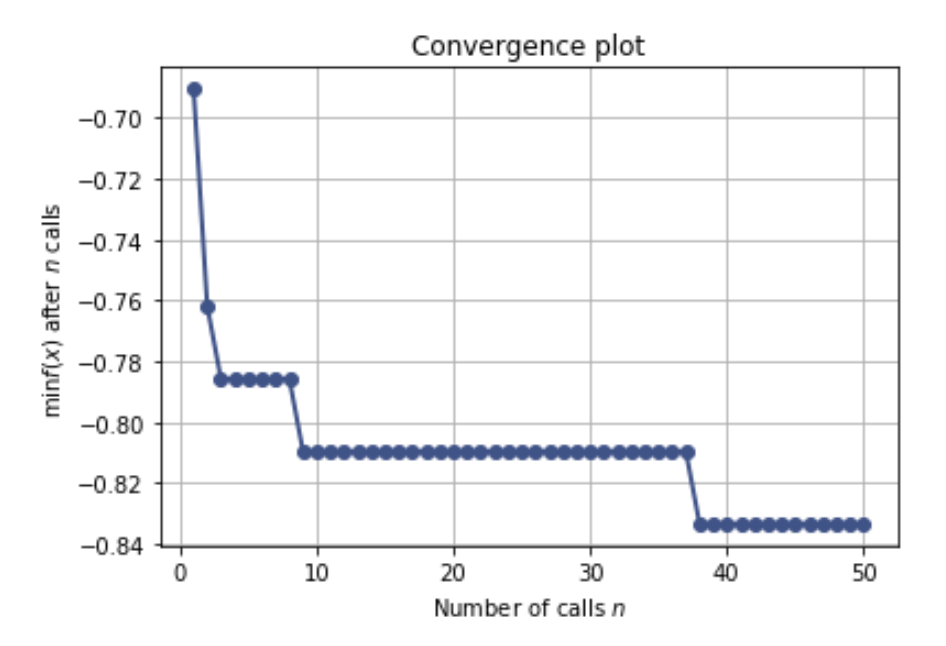
Manual design of NN is a tedious and time consuming process that requires knowledge and some lack. Let’s explore the capabilities of <https://scikit-optimize.github.io/>

### Hyper-parameter turning using Scikit-Optimize

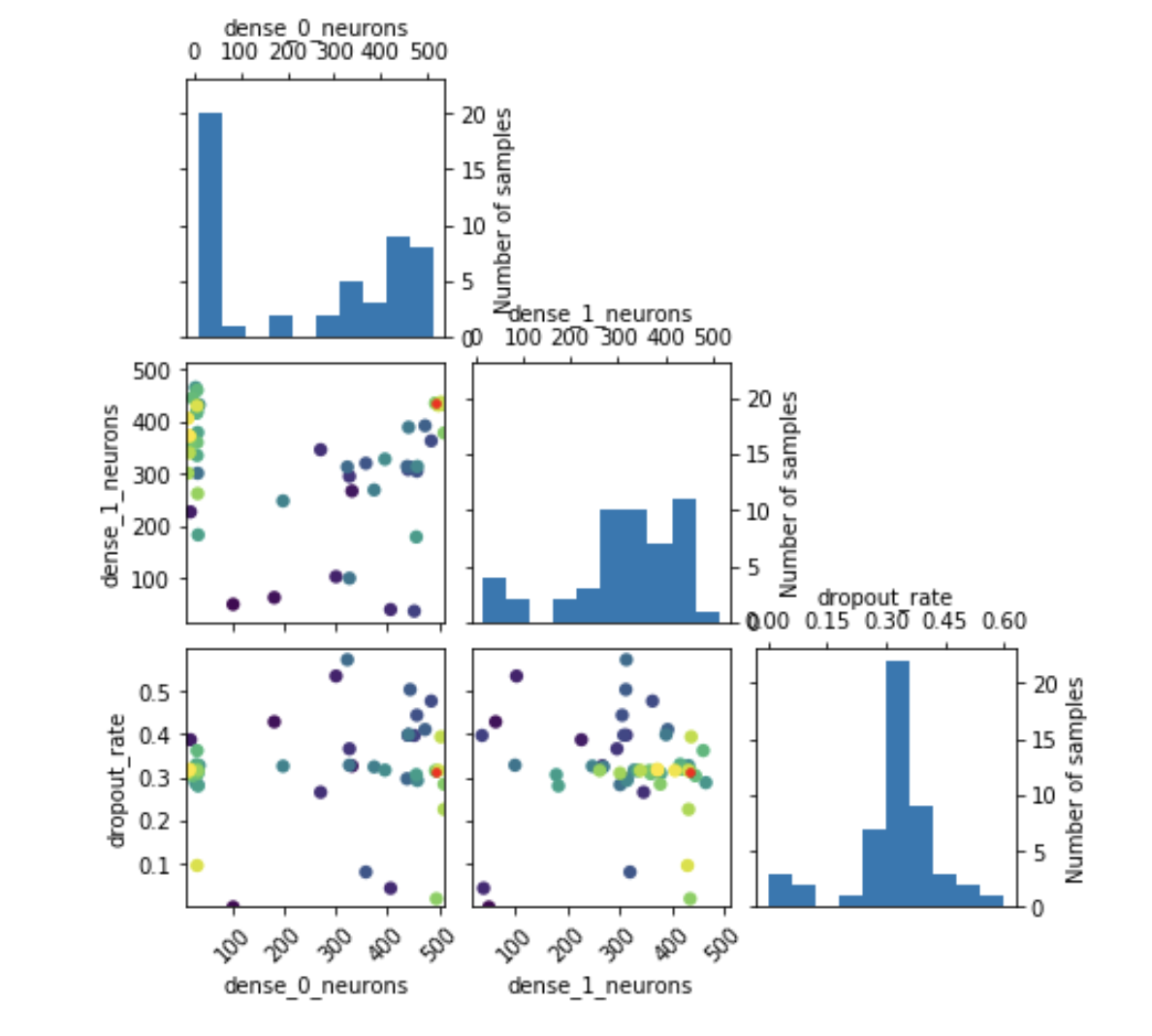
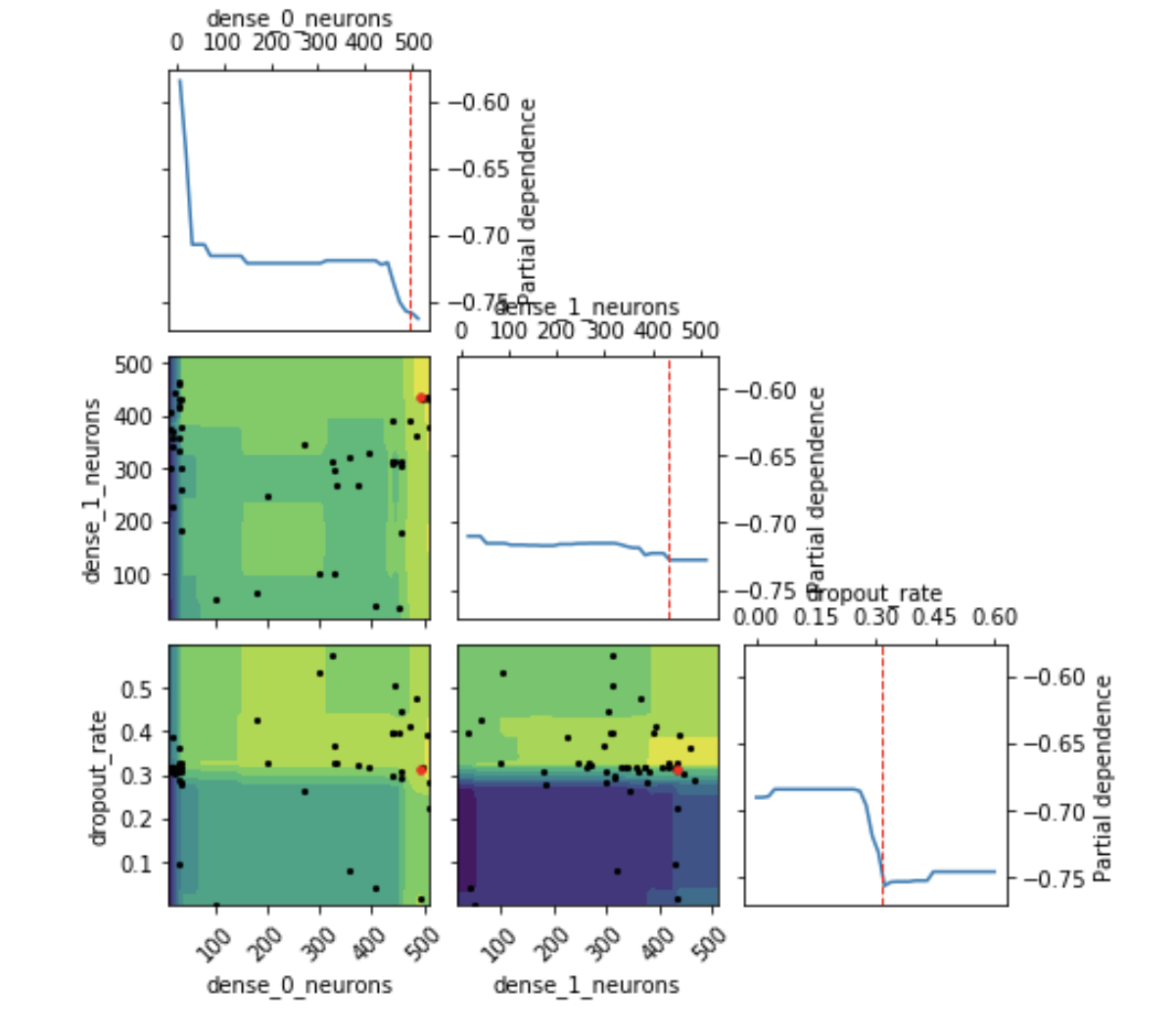
#### The Sonar Dataset

The notebook code is here: <https://github.com/borodark/ie7860/blob/master/HW2%20MLP%20Sonar%20SKOpt.ipynb>

* Convergence Plot



sonar-hyper-conv.png

* Evaluations Plot 
* Objective Partial Dependence Plots 

The suggested topology for two layers network:

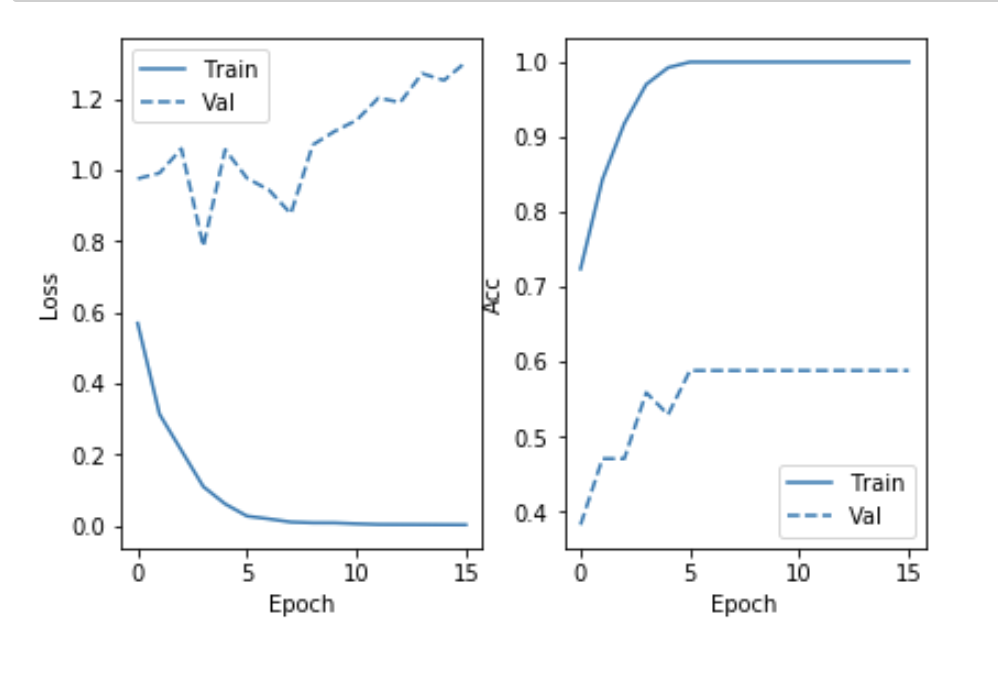
Best score=-0.8333  
Best parameters:  
- dense\_0\_neurons=493  
- dense\_1\_neurons=435  
- dropout\_rate=0.315912

These is rather large size of hydden layers.

#### Running the fit for suggested topology

# Create the model using a specified hyperparameters.  
  
# Get best hyper-parameters  
dense\_0\_neurons=results\_gp.x[0]; dense\_1\_neurons=results\_gp.x[1]; dropout\_rate=results\_gp.x[2]  
  
# Build   
model = get\_model(dense\_0\_neurons, dense\_1\_neurons, dropout\_rate, num\_features, num\_classes)  
  
# Compile the keras model for a specified number of epochs.  
model.compile(loss='binary\_crossentropy',  
 optimizer='adam',  
 metrics=['accuracy'])  
  
# Fit keras model  
history = model.fit(X\_train, y\_train\_one\_hot, epochs=16, batch\_size=8,   
 validation\_split = 0.20, verbose=1)  
  
# Evaluate the model with the eval dataset.  
score = model.evaluate(X\_test, y\_test\_one\_hot,  
 batch\_size=8, verbose=1)  
print('Test loss:', score[0], ' Test accuracy:', score[1])

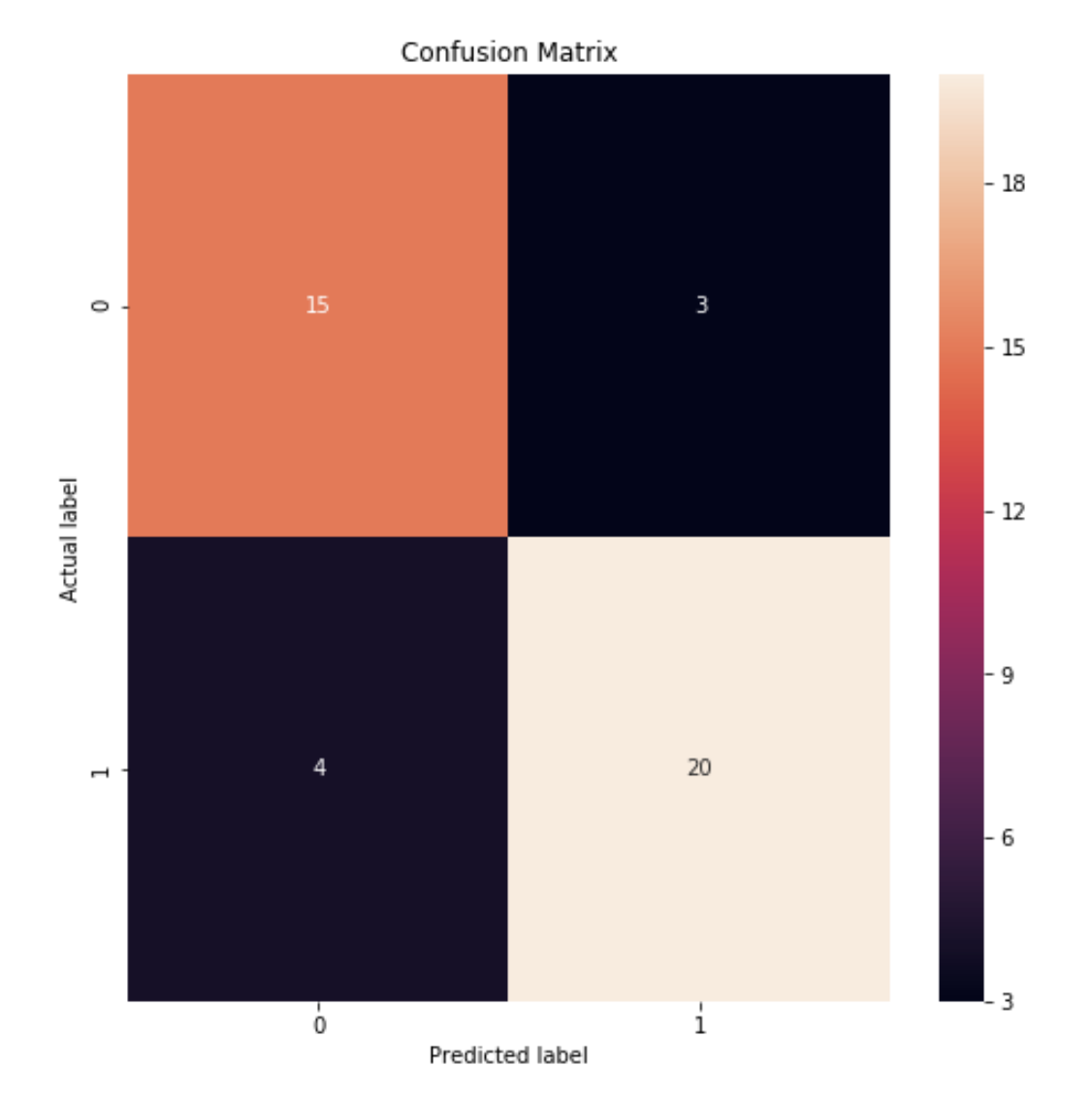
The Losses and accuracy



sonar-hyper-loss-acc.png

loss : 0.5081537847145228  
acc : 0.8333333333333334

The confusion matrix for the generated model



sonar-hyp-cf.png

#### Conclusion

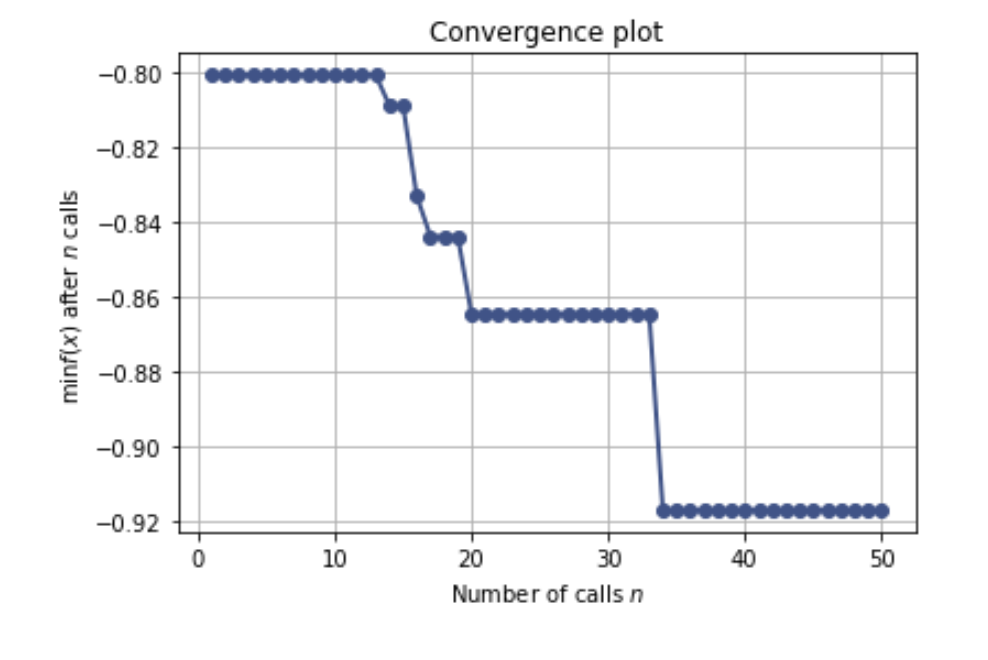
The suggested topology performs better then manually selected 64-16-8 but worse than manually selected 64-16

#### The Thyroid Dataset

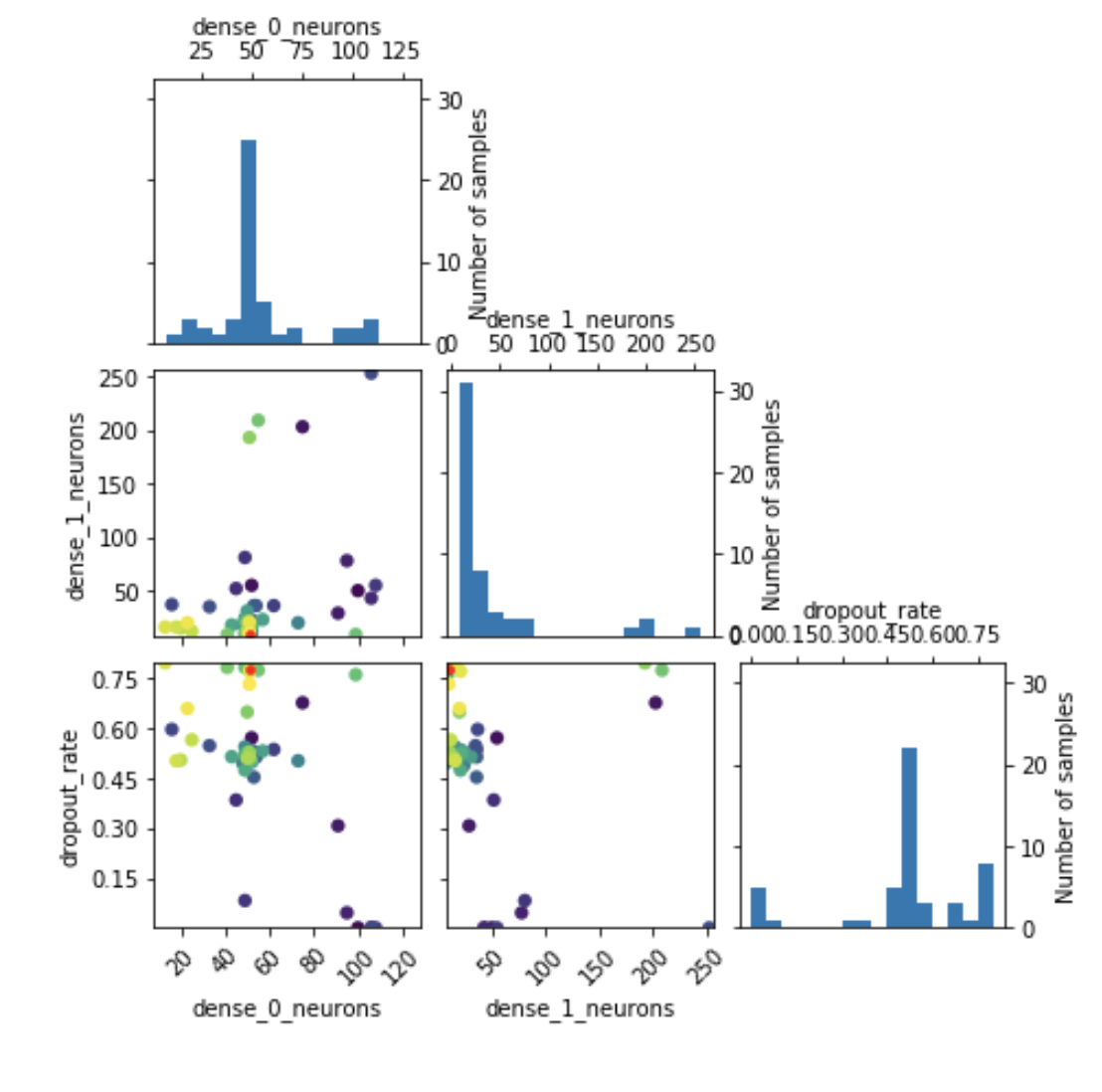
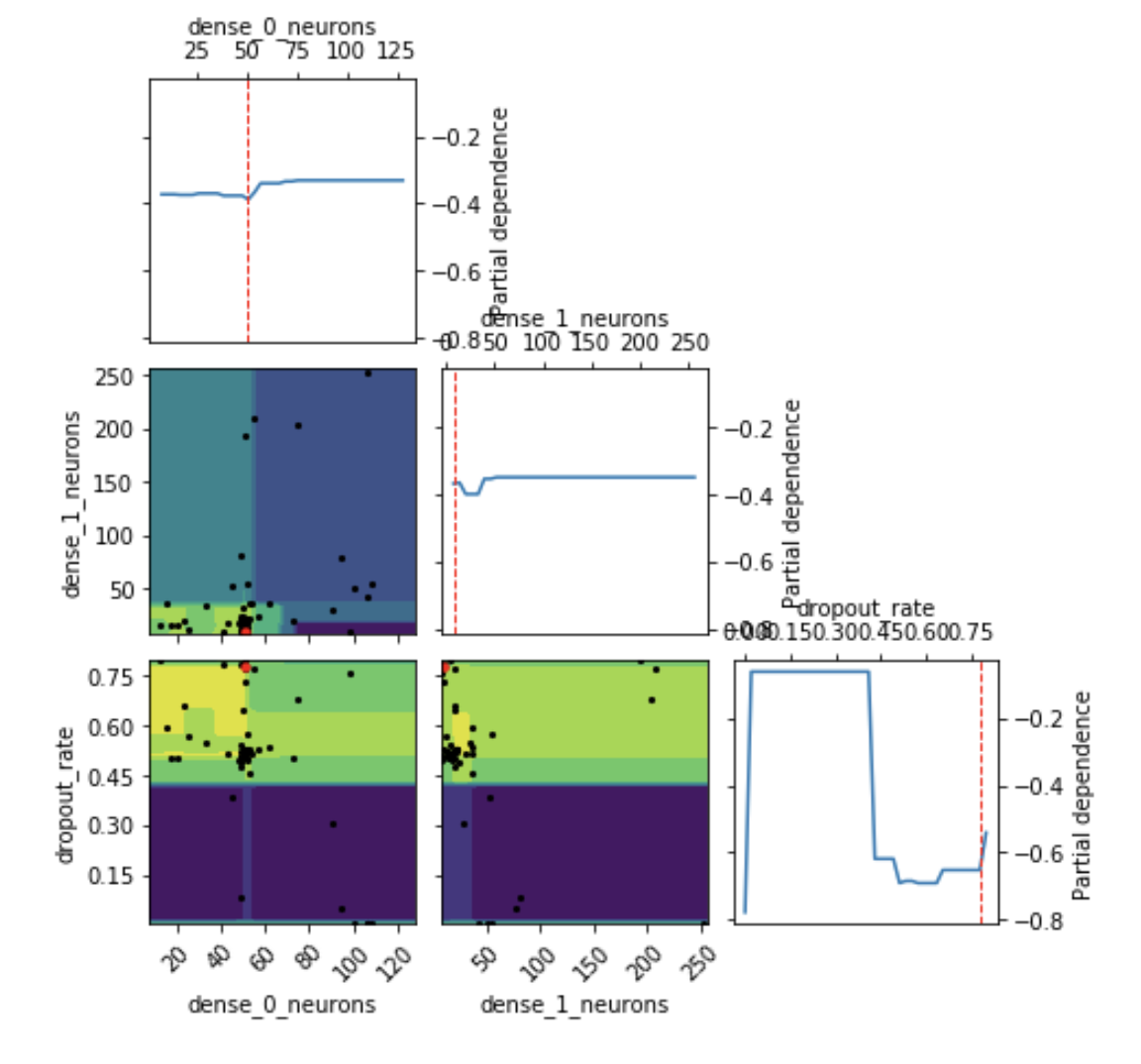
The notebook code is here:

<https://github.com/borodark/ie7860/blob/master/HW2%20MLP%20Thyroid%20SKOpt.ipynb>

* Convergence Plot



thy-hyper-conv.png

* Evaluations Plot 
* Objective Partial Dependence Plots 

The suggested topology for two layers network:

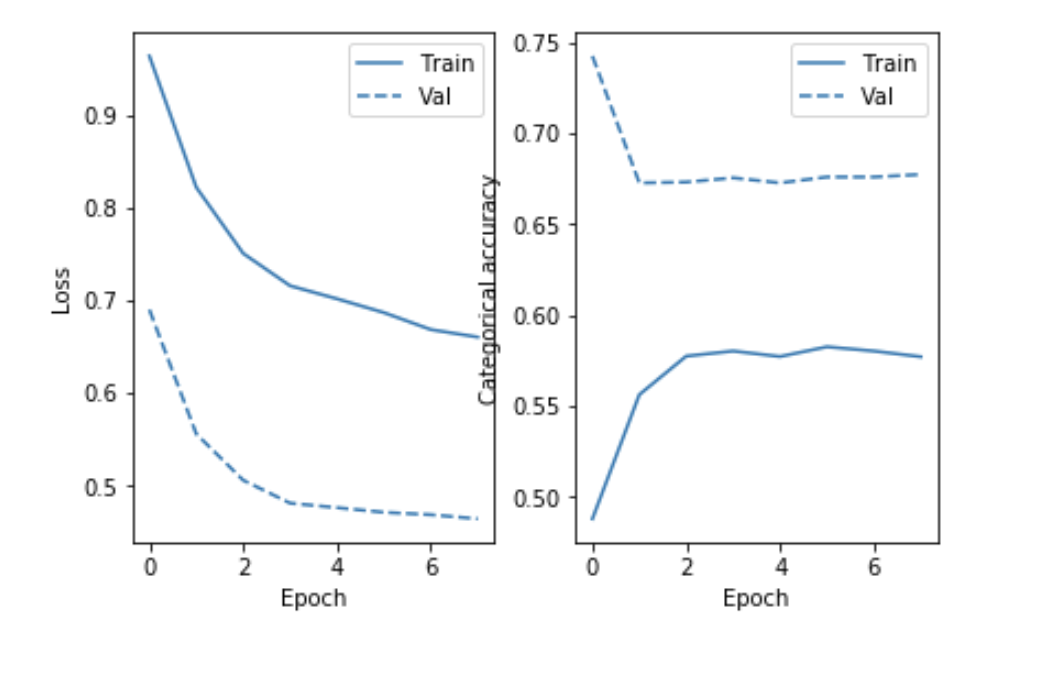
Best score=-0.8466  
Best parameters:  
- dense\_0\_neurons=51  
- dense\_1\_neurons=41  
- dropout\_rate=0.628428

These is rather small size of hydden layers.

#### Running the fit for suggested topology

Train on 8598 samples, validate on 2150 samples  
Epoch 1/8  
8598/8598 [==============================] - 10s 1ms/step - loss: 0.9637 - categorical\_accuracy: 0.4879 - val\_loss: 0.6900 - val\_categorical\_accuracy: 0.7423  
Epoch 2/8  
8598/8598 [==============================] - 1s 132us/step - loss: 0.8220 - categorical\_accuracy: 0.5562 - val\_loss: 0.5559 - val\_categorical\_accuracy: 0.6726  
Epoch 3/8  
8598/8598 [==============================] - 1s 147us/step - loss: 0.7507 - categorical\_accuracy: 0.5773 - val\_loss: 0.5060 - val\_categorical\_accuracy: 0.6730  
Epoch 4/8  
8598/8598 [==============================] - 1s 130us/step - loss: 0.7158 - categorical\_accuracy: 0.5801 - val\_loss: 0.4812 - val\_categorical\_accuracy: 0.6753  
Epoch 5/8  
8598/8598 [==============================] - 1s 165us/step - loss: 0.7017 - categorical\_accuracy: 0.5771 - val\_loss: 0.4764 - val\_categorical\_accuracy: 0.6726  
Epoch 6/8  
8598/8598 [==============================] - 1s 170us/step - loss: 0.6868 - categorical\_accuracy: 0.5825 - val\_loss: 0.4714 - val\_categorical\_accuracy: 0.6758  
Epoch 7/8  
8598/8598 [==============================] - 1s 160us/step - loss: 0.6685 - categorical\_accuracy: 0.5801 - val\_loss: 0.4688 - val\_categorical\_accuracy: 0.6758  
Epoch 8/8  
8598/8598 [==============================] - 1s 149us/step - loss: 0.6604 - categorical\_accuracy: 0.5770 - val\_loss: 0.4643 - val\_categorical\_accuracy: 0.6772  
3428/3428 [==============================] - 0s 56us/step  
Test loss: 0.8766234653416147 Test accuracy: 0.0717619603528025

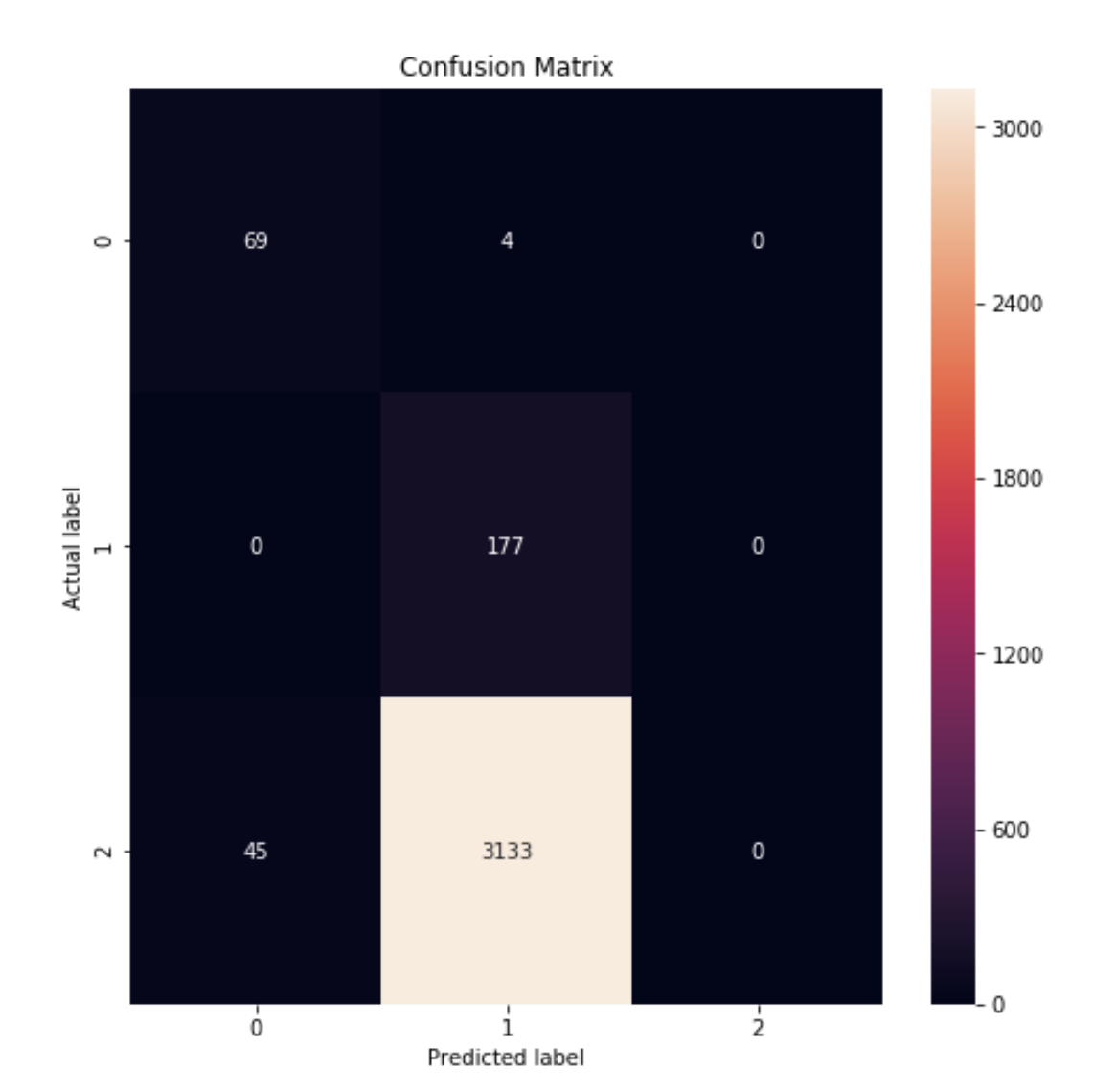
The Losses and accuracy



thy-hyper-loss-acc.png

loss : 0.8766234704187899  
categorical\_accuracy : 0.0717619603484556

The confusion matrix for the generated model



thy-hyp-cf.png

#### Conclusion

The suggested topology collapses on training and fails to perform on the test dataset at all. It is worse than both manually selected network topologies.