Arrhythmia detection using ECG data

Arrhythmia can be detected from a set of irregular heartbeats. These heartbeats produce alterations in the morphology or wave frequency, and all of these alterations can be identified by the ECG exam.

Data set: We have taken the ECG data from Kaggle(HYPERLINK "https://www.kaggle.com/shayanfazeli/heartbeat"https://www.kaggle.com/shayanfazeli/heartbeat)

This dataset is composed of two collections of heartbeat signals derived from two famous datasets in heartbeat classification, the MIT-BIH Arrhythmia Dataset and The PTB Diagnostic ECG Database. The number of samples in both collections is large enough for training a deep neural network.

Arrhythmia Dataset

- Number of Samples: 109446
- Number of Categories: 5
- Sampling Frequency: 125Hz
- Data Source: Physionet's MIT-BIH Arrhythmia Dataset
- Classes: ['N': 0, 'S': 1, 'V': 2, 'F': 3, 'Q': 4]

The PTB Diagnostic ECG Database

- Number of Samples: 14552
- Number of Categories: 2
- Sampling Frequency: 125Hz
- Data Source: Physionet's PTB Diagnostic Database

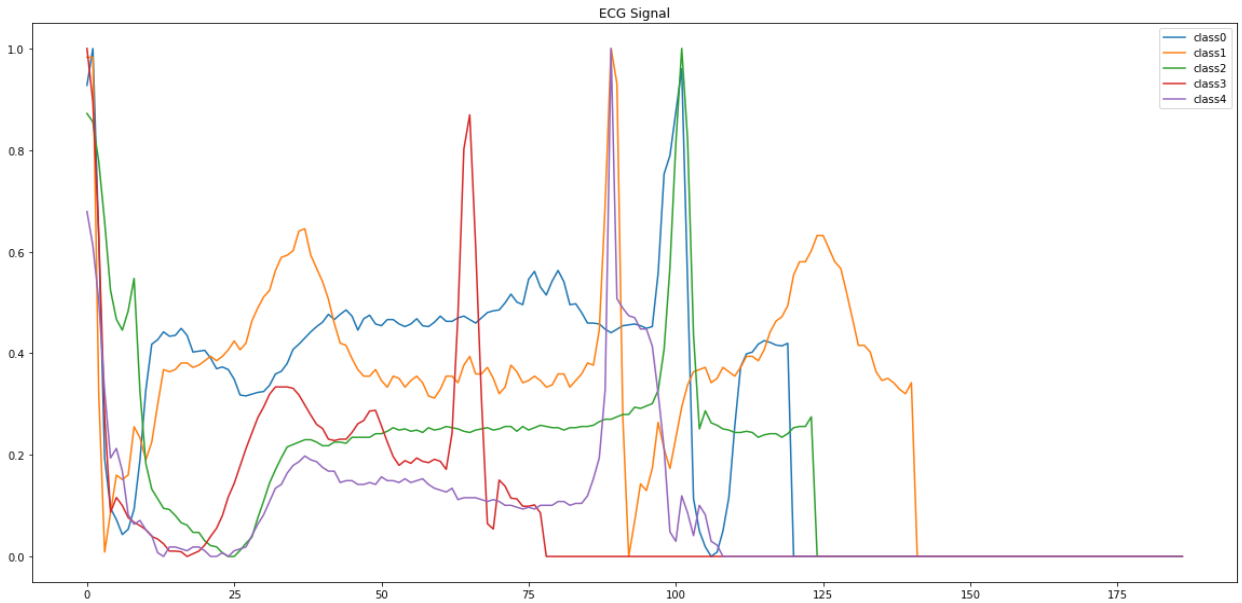
In this experiment we used **Arrhythmia Dataset.** This data set contains train and test sets.

mitbih_train: 87554*188

mitbih_test:21892*188

```
Type    Count
0.0    72471
4.0     6431
2.0     5788
1.0     2223
3.0      641
Train shape :  (87554, 188)
Test shape :  (21892, 188)
```

The below plot represents how the classes spread over the data.



Feature Engineering: We extracted below features from the data.

- `peaks`
- `height`
- `width`
- `prominence`
- `arg_min`
- `arg_max`

Models :

Machine learning models for binary classification :

Initially we tried with machine learning models. We started with basic model as we converted the problem into binary classification.

As there is some linear dependency between the target and features logistic regression model performance is good, still we can improve if we can go to non linear models. So we tried several non linear models like SVM, Decision tree, Random forest and we also tried with statistical model Naïve bayes. Some models of machine learning are:

1. **Naive Bayes :**

   Confusion matrix

	precision	recall	f1-score
0	0.89	0.88	0.88
1	0.45	0.49	0.47

Accuracy of Naive Bayes model is : 0.8099305682441075

2. **Stochastic Gradient Descent :**

   Confusion matrix

	precision	recall	f1-score
0	0.90	0.98	0.94
1	0.82	0.48	0.60

Accuracy of Stochastic Gradient Descent model is : 0.8912844874840125

3. **Logistic Regression :**

   Confusion matrix

	precision	recall	f1-score
0	0.91	0.98	0.94
1	0.84	0.55	0.67

Accuracy of Logistic Regression model is : 0.9047597295815824

4. **Decision Binary Tree :**

   Confusion matrix

	precision	recall	f1-score
0	0.96	0.99	0.98
1	0.94	0.81	0.87

Accuracy of Decision Binary Tree model is : 0.9587520555454048

5. **Random Forest :**

Confusion matrix

	precision	recall	f1-score
0	0.97	0.99	0.98
1	0.96	0.84	0.90

Accuracy of Random Forest model is : 0.9667001644436324

6. **K-Nearest Neighbors :**

Confusion matrix

	precision	recall	f1-score
0	0.97	0.99	0.98
1	0.96	0.86	0.91

Accuracy of KNN model is : 0.9695322492234606

7. **Support Vector Machine :**

Confusion matrix

	precision	recall	f1-score
0	0.97	1.00	0.98
1	0.96	0.84	0.91

Accuracy of SVM model is : 0.9700803946647177

The below table will show you performance of each model

	Model	Accuracy	Precision	Recall	F1_score
1	Support Vector Machine	0.970080	0.970398	0.970080	0.969161
2	K-Nearest Neighbors	0.969532	0.969364	0.969532	0.968826
3	Random Forest	0.966700	0.966567	0.966700	0.965797
4	Decision Tree	0.958752	0.958225	0.809931	0.957552
5	Logistic Regression	0.904760	0.900424	0.904760	0.896620
6	Stochastic Gradient Descent	0.891284	0.885269	0.891284	0.879306
7	Naive Bayes	0.809931	0.816899	0.809931	0.813189

We can clearly see that the SVM performs better than all other models. This may be because the data is in higher dimension and as we used RBF kernel in SVM to transform the non linear dependencies to higher representation in such a way that it can form a hyperplane to separate the classes. There is one more reason that SVM will always gives the global minima.

Deep Learning models :

DL helps when the data have higher dimensions and large. It extracts the features internally.

In this experiment we tried with simple Multilayer Perceptron with different hyper parameters.

Here we have applied some feature transformations for the scaling of data. Feature transform helps in speed up to find the global minima. We used standard scalar.

The formula for standard scalar:

x_i= (x_i-mean) / standard deviation

After applying feature transformation, the data will transform to normal distribution with unit variance. We used different type of models:

1. **Softmax classifier**

In this model we used softmax as activation function and the model architecture as given below:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 2)                 376
=================================================================
Total params: 376
Trainable params: 376
Non-trainable params: 0
_________________________________________________________________
```

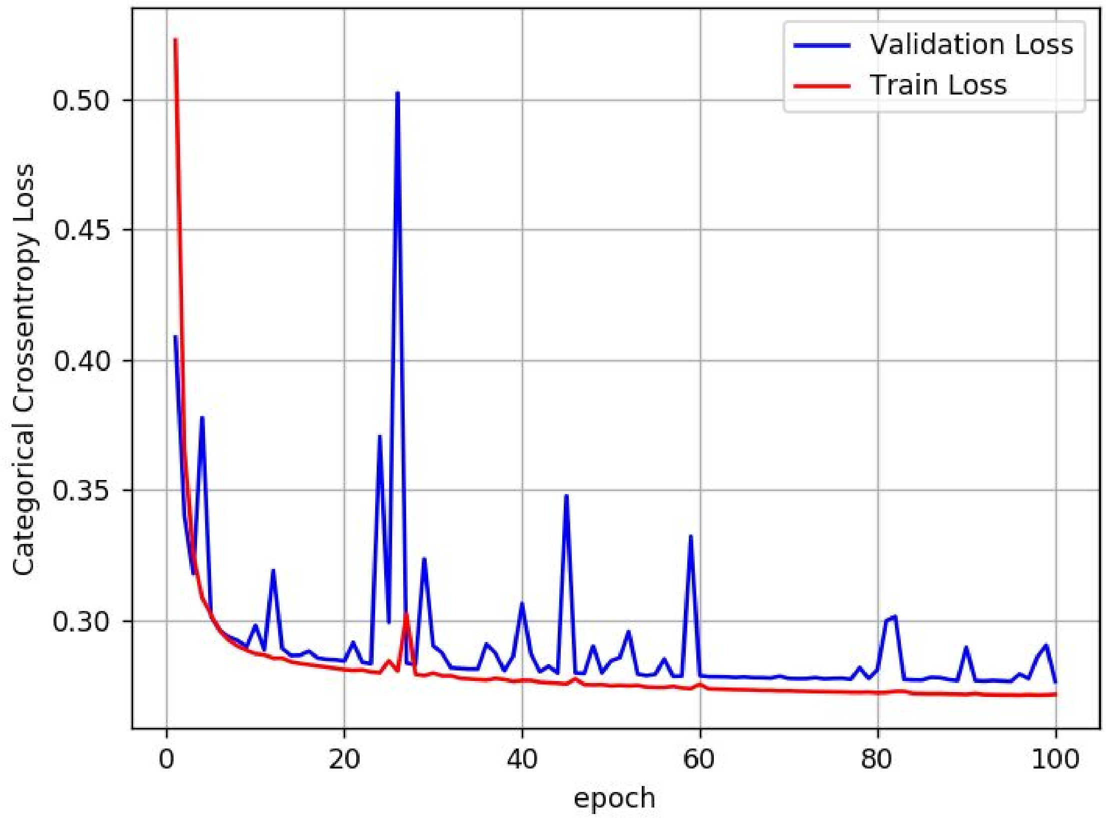
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94     18118
           1       0.85      0.54      0.66      3774

    accuracy                           0.90     21892
   macro avg       0.88      0.76      0.80     21892
weighted avg       0.90      0.90      0.90     21892
```

Test score: 0.27609830279575065

Test accuracy: 0.904714047908783

Loss Plot



2. **Binary Classifier with Sigmoid and SGD Optimizer**

In this model we used sigmoid as activation function and Stochastic Gradient Descent (SGD) as optimizer. The architecture of this model is given below:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               24064
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
```

```
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 130
=================================================================
Total params: 32,450
Trainable params: 32,450
Non-trainable params: 0
_________________________________________________________________
```

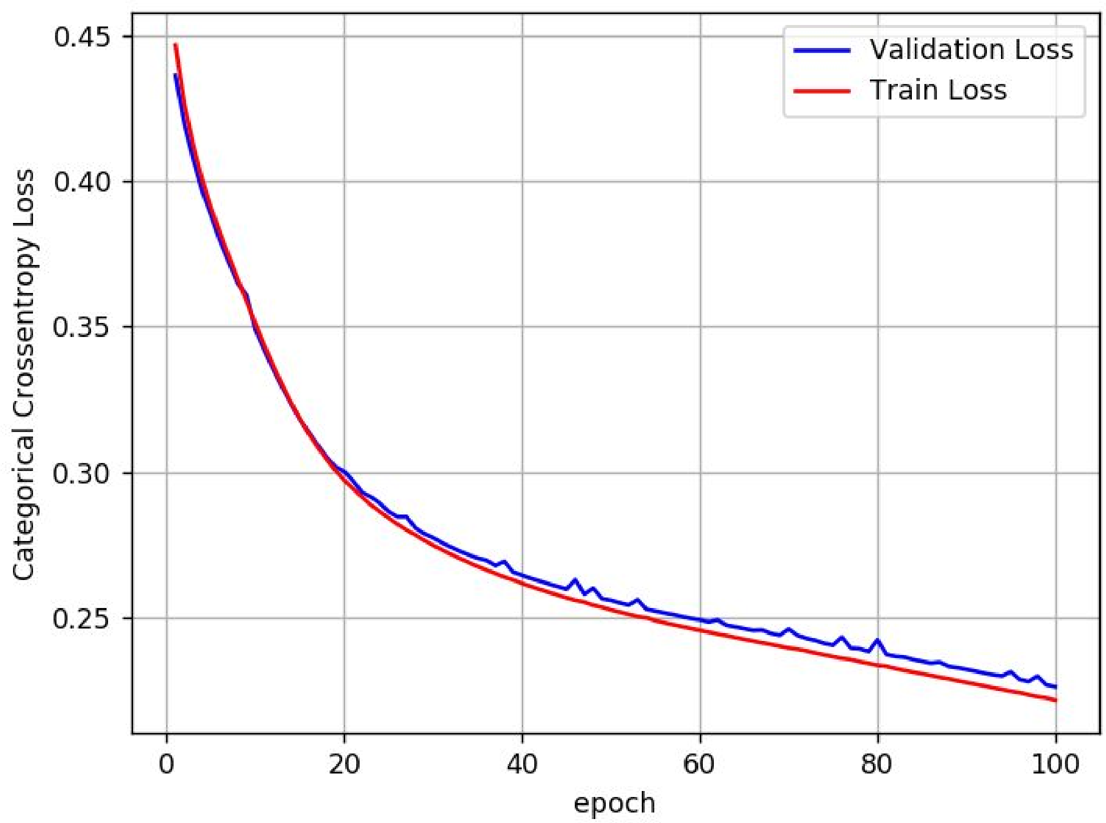
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     18118
           1       0.92      0.64      0.76      3774

    accuracy                           0.93     21892
   macro avg       0.93      0.82      0.86     21892
weighted avg       0.93      0.93      0.92     21892
```

Test score: 0.22617576710598006

Test accuracy: 0.928878128528595

Loss Plot



3. Binary Classifier with Sigmoid and ADAM Optimizer

In this model we used sigmoid as activation function and Adaptive moment estimation as optimizer. **Adam** combines the best properties of the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. The architecture of this model is given below:

```
Model: "sequential_1"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================
```

```
dense_1 (Dense)              (None, 128)               24064
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 130
=================================================================
Total params: 32,450
Trainable params: 32,450
Non-trainable params: 0
_________________________________________________________________
```

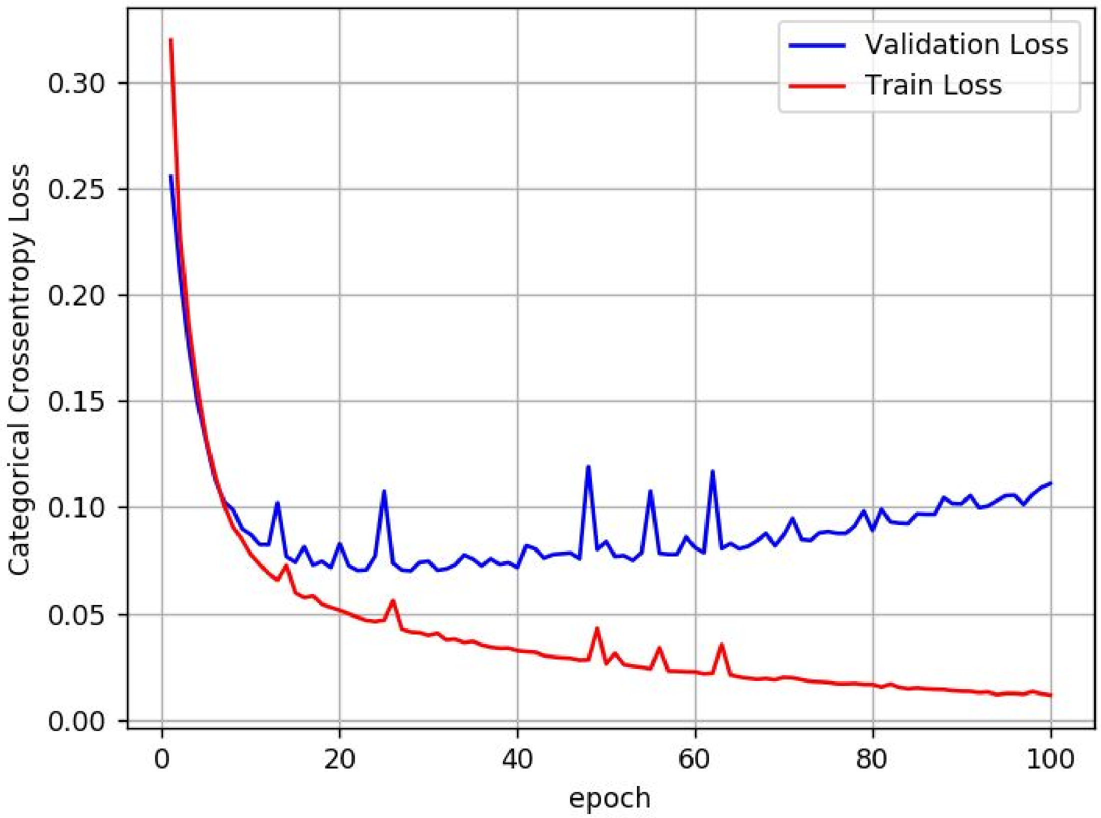
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     18118
           1       0.95      0.93      0.94      3774

    accuracy                           0.98     21892
   macro avg       0.97      0.96      0.96     21892
weighted avg       0.98      0.98      0.98     21892
```

Test score: 0.11119293791238763

Test accuracy: 0.9797186255455017

Loss Plot



4. **Binary Classifier with ReLU and SGD Optimizer**

In this model we used Rectified linear units (ReLU) as activation function and Stochastic Gradient Descent (SGD) as optimizer. ReLU used would be faster and it is more biological inspired. It gives sparsity and reduced likelihood of vanishing gradient. The architecture of this model is given below:

```
Model: "sequential_5"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_13 (Dense)             (None, 128)               24064

_________________________________________________________________
```

```
dense_14 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_15 (Dense)             (None, 2)                 130
=================================================================
Total params: 32,450

Trainable params: 32,450

Non-trainable params: 0
_________________________________________________________________
```

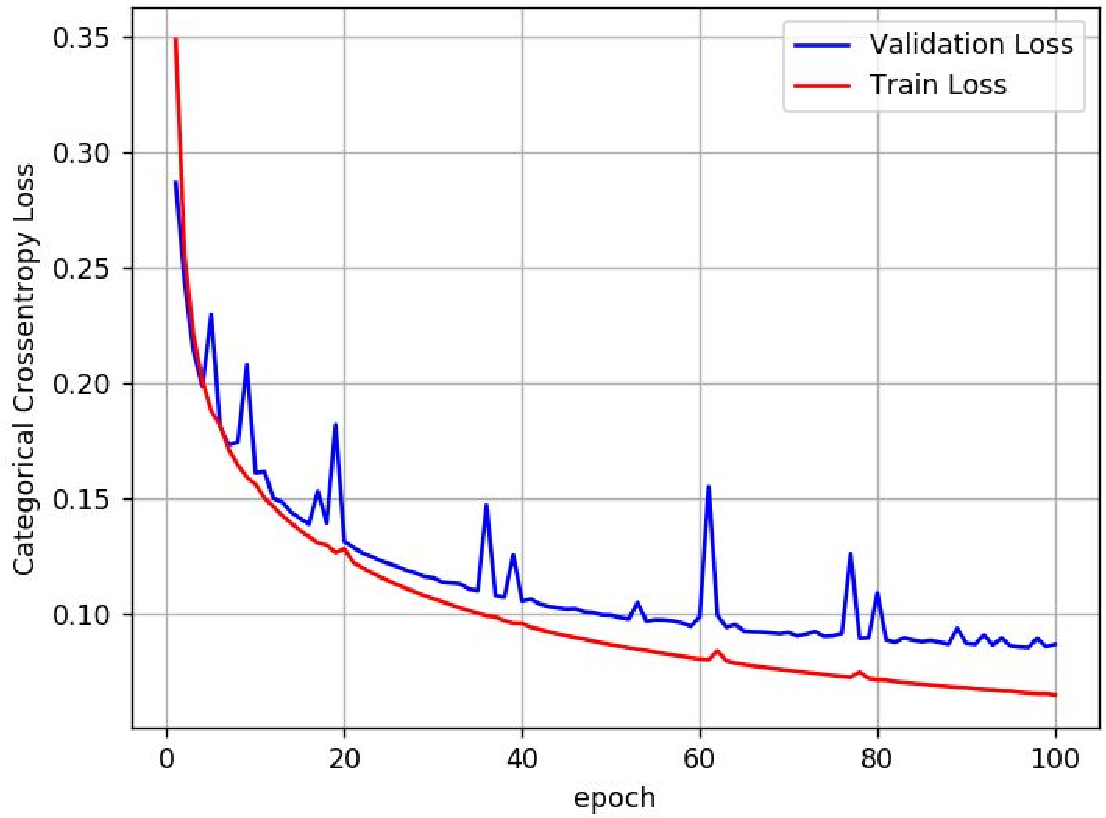
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     18118
           1       0.96      0.87      0.92      3774

    accuracy                           0.97     21892
   macro avg       0.97      0.93      0.95     21892
weighted avg       0.97      0.97      0.97     21892
```

Test score: 0.08665858121973743

Test accuracy: 0.9727754592895508

Loss Plot



5. Binary Classifier with ReLU and ADAM Optimizer

In this model we used Rectified linear units (ReLU) as activation function and Adaptive moment estimation as optimizer. The architecture of this model is given below:

```
Model: "sequential_1"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_1 (Dense)              (None, 128)               24064

_________________________________________________________________

dense_2 (Dense)              (None, 64)                8256
```

```
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 130
=================================================================
Total params: 32,450
Trainable params: 32,450
Non-trainable params: 0
_________________________________________________________________
None
```

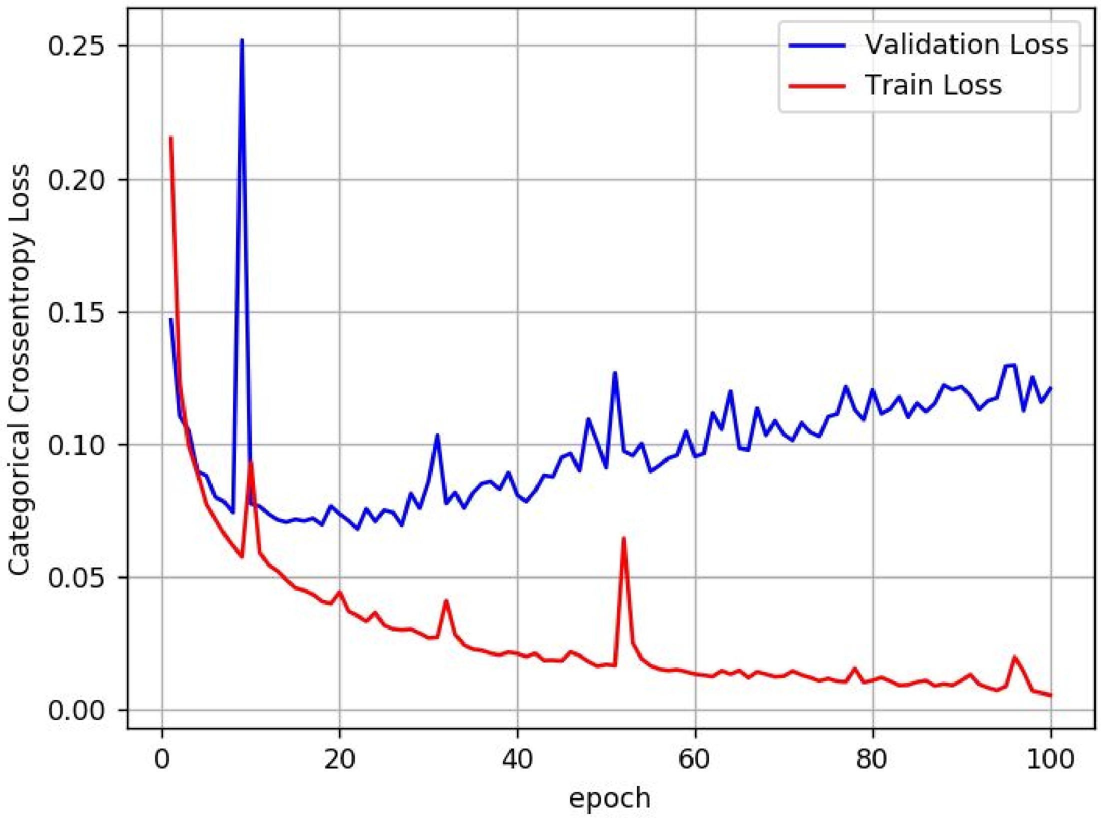
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     18118
           1       0.96      0.94      0.95      3774

    accuracy                           0.98     21892
   macro avg       0.97      0.97      0.97     21892
weighted avg       0.98      0.98      0.98     21892
```

Test score: 0.12087892898057417

Test accuracy: 0.9822766184806824

Loss Plot



6. **Binary Classifier with sigmoid, Batch-Norm and Adam Optimizer**

In this model we used sigmoid as activation function, Adaptive moment estimation as optimizer (ADAM) and we applied batch normalization on hidden layers. The architecture of this model is given below:

```
Model: "sequential_1"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_1 (Dense)              (None, 128)               24064

_________________________________________________________________

batch_normalization_1 (Batch (None, 128)               512
```

```
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256      
_________________________________________________________________
batch_normalization_2 (Batch (None, 64)                256       
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 130       
=================================================================
Total params: 33,218
Trainable params: 32,834
Non-trainable params: 384
_________________________________________________________________
```

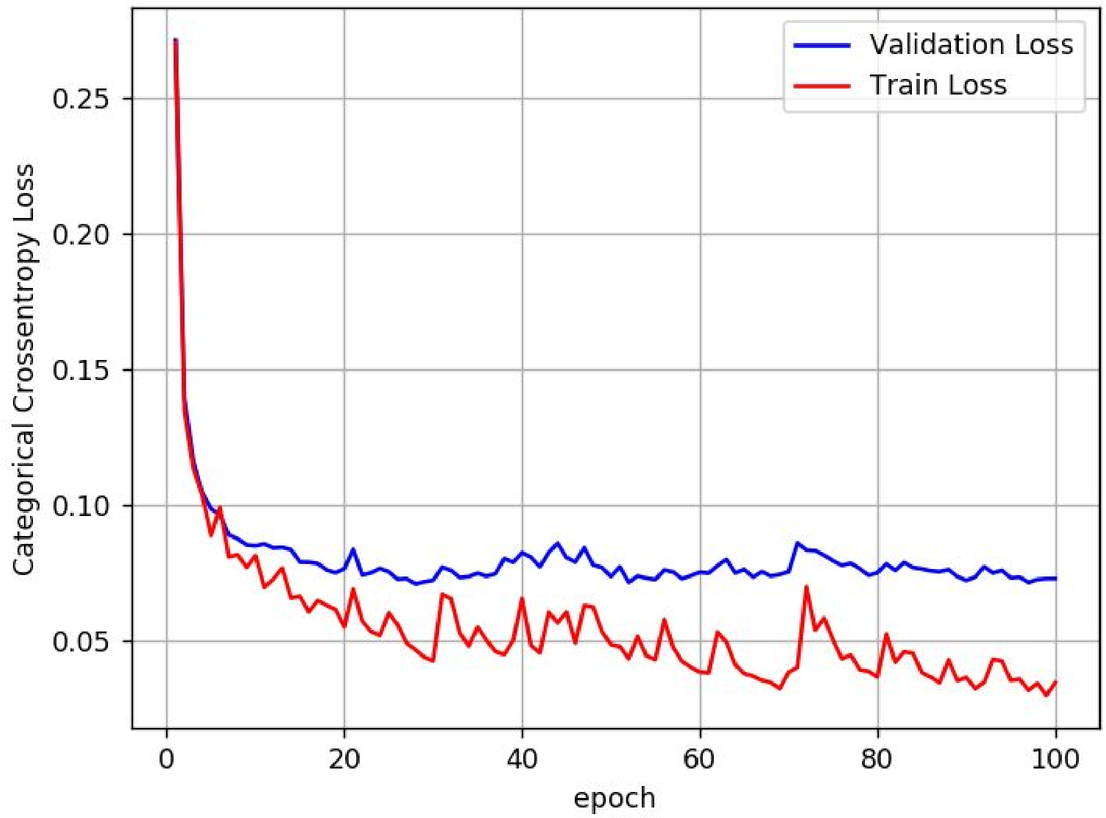
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     18118
           1       0.96      0.93      0.94      3774

    accuracy                           0.98     21892
   macro avg       0.97      0.96      0.96     21892
weighted avg       0.98      0.98      0.98     21892
```

Test score: 0.07275567385506736

Test accuracy: 0.979992687702179

Loss Plot



7. Binary Classifier with sigmoid, Dropout and Adam Optimizer

In this model we used sigmoid as activation function and Adaptive moment estimation as optimizer (ADAM) and we also used dropout. A fully connected layer occupies most of the parameters, and hence, neurons develop co-dependency amongst each other during training which curbs the individual power of each neuron leading to over-fitting in training data. That's why we used dropout approach for regularization in neural networks which helps reducing interdependent learning amongst the neurons. The architecture of this model is given below:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
dense_1 (Dense)              (None, 128)               24064
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
batch_normalization_2 (Batch (None, 64)                256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 130
=================================================================
Total params: 33,218
Trainable params: 32,834
Non-trainable params: 384
_________________________________________________________________
```

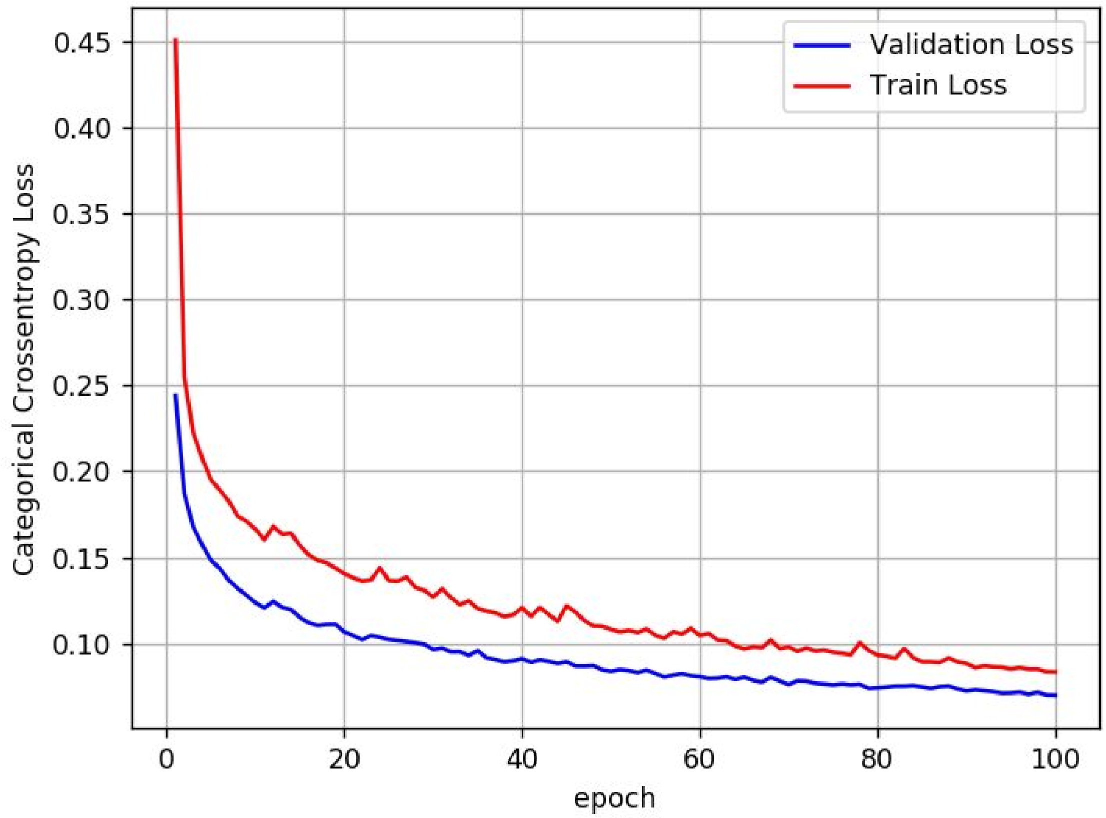
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     18118
           1       0.97      0.90      0.94      3774

    accuracy                           0.98     21892
   macro avg       0.98      0.95      0.96     21892
weighted avg       0.98      0.98      0.98     21892
```

Test score: 0.06966984133317279

Test accuracy: 0.9784396290779114

Loss Plot



8. Binary Classifier with ReLU, Batch Normalization, Dropout and Adam Optimizer

In this model we used ReLU as activation function and Adaptive moment estimation as optimizer (ADAM) and we also used batch normalization and dropout. We used Batch normalization to make sure that every batch of the data should be in the normal distribution. The architecture of this model is given below:

```
Model: "sequential_1"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_1 (Dense)              (None, 128)               24064

_________________________________________________________________
```

```
batch_normalization_1 (Batch (None, 128)              512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)              0
_________________________________________________________________
dense_2 (Dense)              (None, 64)               8256
_________________________________________________________________
batch_normalization_2 (Batch (None, 64)               256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                130
=================================================================
Total params: 33,218
Trainable params: 32,834
Non-trainable params: 384
_________________________________________________________________
None
```

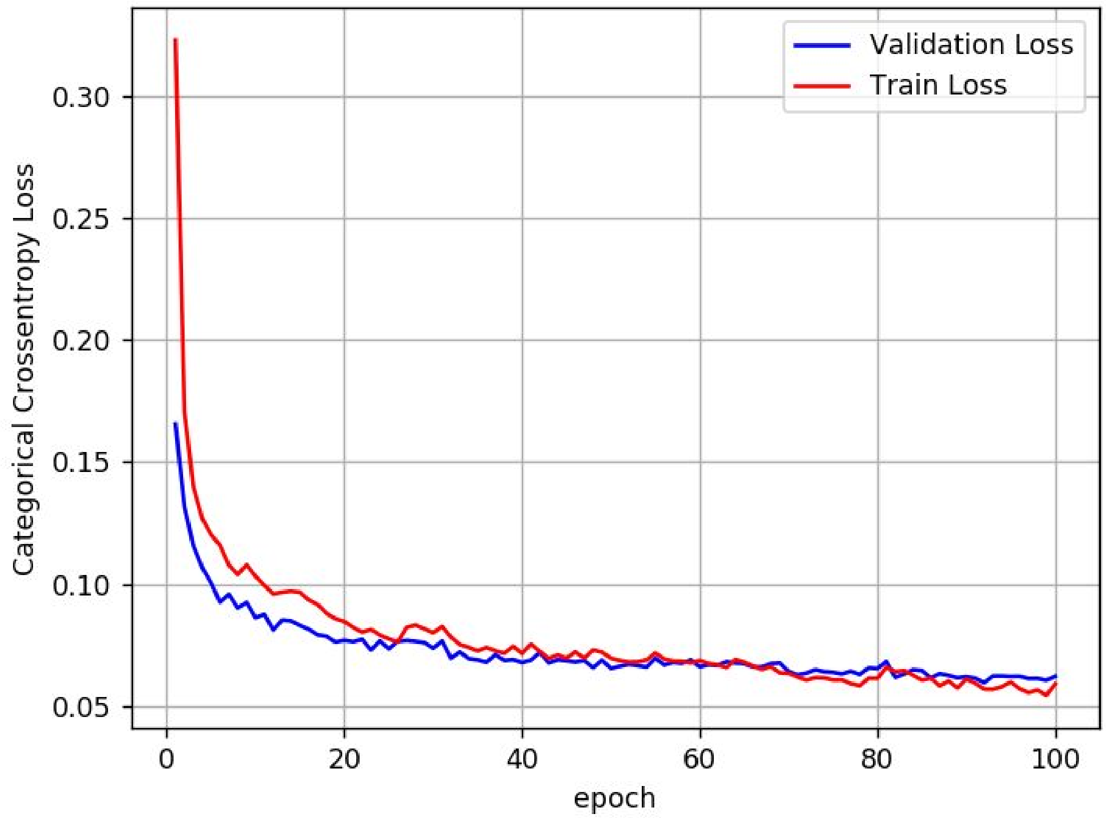
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     18118
           1       0.98      0.91      0.95      3774

    accuracy                           0.98     21892
   macro avg       0.98      0.95      0.97     21892
weighted avg       0.98      0.98      0.98     21892
```

Test score: 0.062071498292025705

Test accuracy: 0.9820025563240051

Loss Plot



Long short-term memory model binary classification:

Next we started experimenting the Long short-term memory (LSTM). As the ECG data is the time series, we used LSTM. LSTM performs better with time series because of its property maintaining temporal dependency. It is an RNN variant where the input will carry all the time steps. This model gave better performance than all other models.

Confusion matrix

	precision	recall	f1-score	support
0	0.99	0.99	0.99	18118
1	0.96	0.94	0.95	3774
accuracy			0.98	21892
macro avg	0.98	0.97	0.97	21892
weighted avg	0.98	0.98	0.98	21892

Accuracy : 0.98

Multi-class classification

We have also tried to do multiclass classification. Here we tried to all 5 classes in which 4 are disease, 1 is no disease.

We tried all the models which we used for binary classification.

- Machine learning models
- Deep learning models (Multi Layer Perceptron)
- LSTM model

Machine Learning Models :

Below table represents the Machine learning models performance.

	Model	Accuracy	Precision	Recall	F1_score
1	Support Vector Machine	0.969075	0.968616	0.969075	0.966878
2	K-Nearest Neighbors	0.966197	0.965173	0.966197	0.964360
3	Random Forest	0.959163	0.957615	0.959163	0.955000
4	Decision Binary Tree	0.953407	0.950803	0.953407	0.949715
5	Logistic Regression	0.907911	0.896475	0.907911	0.891540
6	Stochastic Gradient Descent	0.843230	0.834042	0.843202	0.795375
7	Naive Bayes	0.418006	0.726515	0.418006	0.514668

Deep Learning models for multi-class classification:

DL helps when the data have higher dimensions and large. It extracts the features internally.

In this experiment we tried with simple Multilayer Perceptron with different hyper parameters.

Here we have applied some feature transformations for the scaling of data. Feature transform helps in speed up to find the global minima. We used standard scalar.

The formula for standard scalar:

x_i= (x_i-mean) / standard deviation

After applying feature transformation, the data will transform to normal distribution with unit variance. We used different types of models:

1. **Softmax multi classifier**

In this model we used softmax as activation function and the model architecture as given below:

```
Model: "sequential_3"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_3 (Dense)              (None, 5)                 940

=================================================================

Total params: 940

Trainable params: 940

Non-trainable params: 0

_________________________________________________________________
```

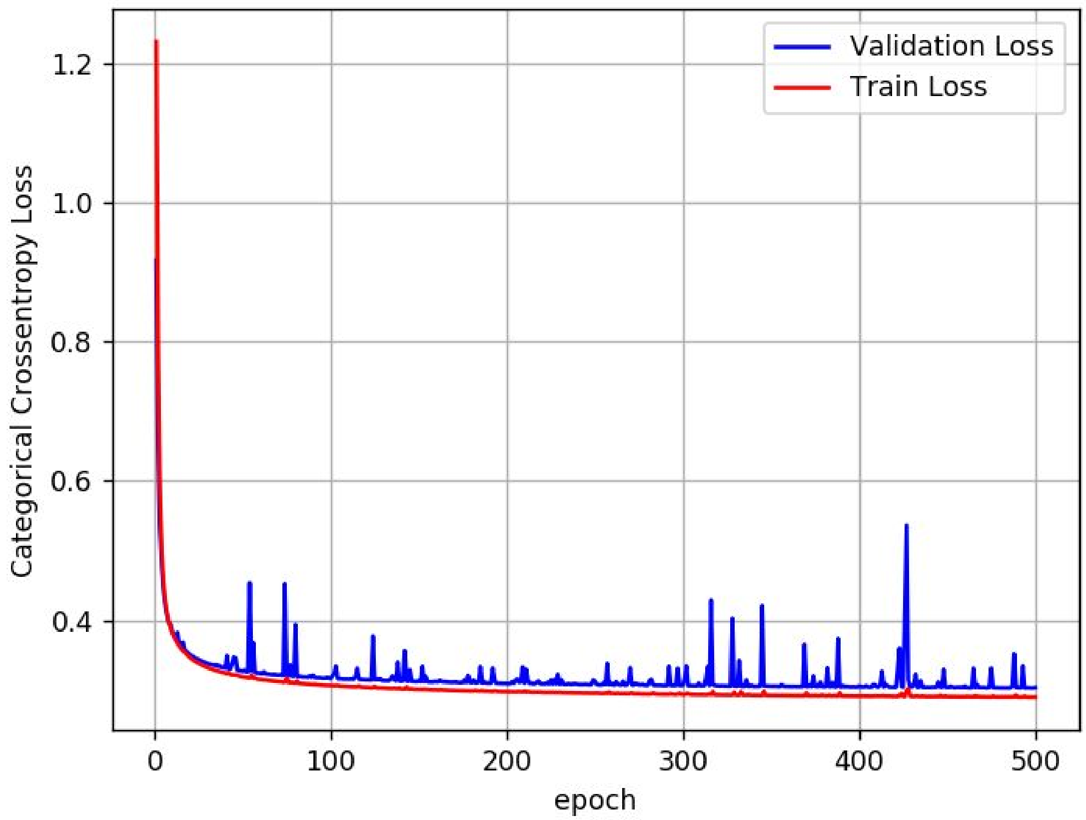
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95     18118
           1       0.83      0.39      0.53       556
           2       0.61      0.36      0.46      1448
           3       0.66      0.22      0.33       162
           4       0.97      0.87      0.92      1608

    accuracy                           0.91     21892
   macro avg       0.80      0.56      0.64     21892
weighted avg       0.90      0.91      0.90     21892
```

Test score: 0.30287555285395773

Test accuracy: 0.9120683073997498

Loss Plot



2. **Multi classifier with Sigmoid and SGD Optimizer**

In this model we used sigmoid as activation function and Stochastic Gradient Descent (SGD) as optimizer

```
Model: "sequential_1"
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               24064
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 325
=================================================================
Total params: 32,645
Trainable params: 32,645
Non-trainable params: 0
_________________________________________________________________
```

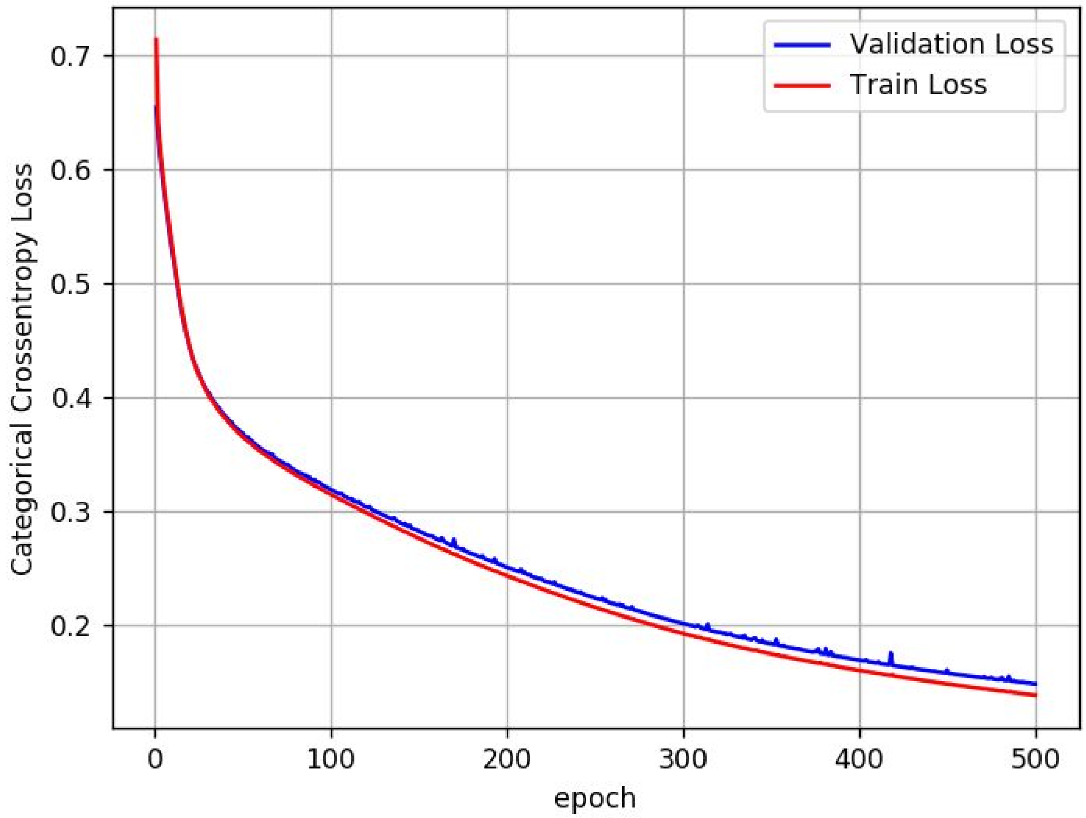
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     18118
           1       0.94      0.52      0.67       556
           2       0.91      0.84      0.87      1448
           3       0.68      0.25      0.37       162
           4       0.98      0.90      0.94      1608

    accuracy                           0.96     21892
   macro avg       0.90      0.70      0.77     21892
weighted avg       0.96      0.96      0.96     21892
```

Test score: 0.14833274375046276

Test accuracy: 0.9604878425598145

Loss Plot



3. **Multi classifier with Sigmoid and ADAM Optimizer**

In this model we used sigmoid as activation function and Adaptive moment estimation as optimizer. **Adam** combines the best properties of the Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.

```
Model: "sequential_1"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_1 (Dense)              (None, 128)               24064


_________________________________________________________________

dense_2 (Dense)              (None, 64)                8256
```

```
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 325
=================================================================
Total params: 32,645

Trainable params: 32,645

Non-trainable params: 0
_________________________________________________________________
```

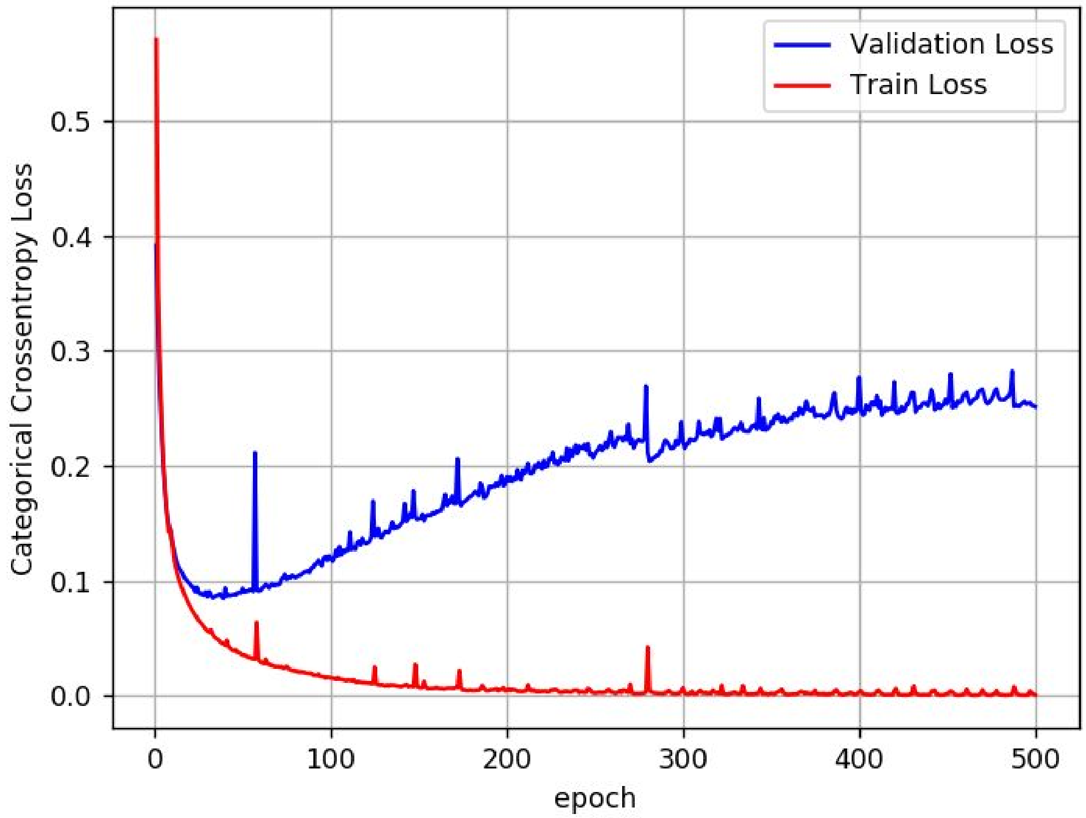
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     18118
           1       0.81      0.73      0.76       556
           2       0.94      0.94      0.94      1448
           3       0.84      0.77      0.80       162
           4       0.98      0.97      0.98      1608

    accuracy                           0.98     21892
   macro avg       0.91      0.88      0.89     21892
weighted avg       0.98      0.98      0.98     21892
```

Test score: 0.2513332234385764

Test accuracy: 0.977160632610321

Loss Plot



4. **Multi classifier with ReLU and SGD Optimizer**

In this model we used Rectified linear units (ReLU) as activation function and Stochastic Gradient Descent (SGD) as optimizer. ReLU used would be faster and it is more biological inspired. It gives sparsity and reduced likelihood of vanishing gradient.

```
Model: "sequential_1"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_1 (Dense)              (None, 128)               24064

_________________________________________________________________

dense_2 (Dense)              (None, 64)                8256

_________________________________________________________________
```

```
dense_3 (Dense)              (None, 5)                 325
=================================================================
Total params: 32,645
Trainable params: 32,645
Non-trainable params: 0
_________________________________________________________________
```

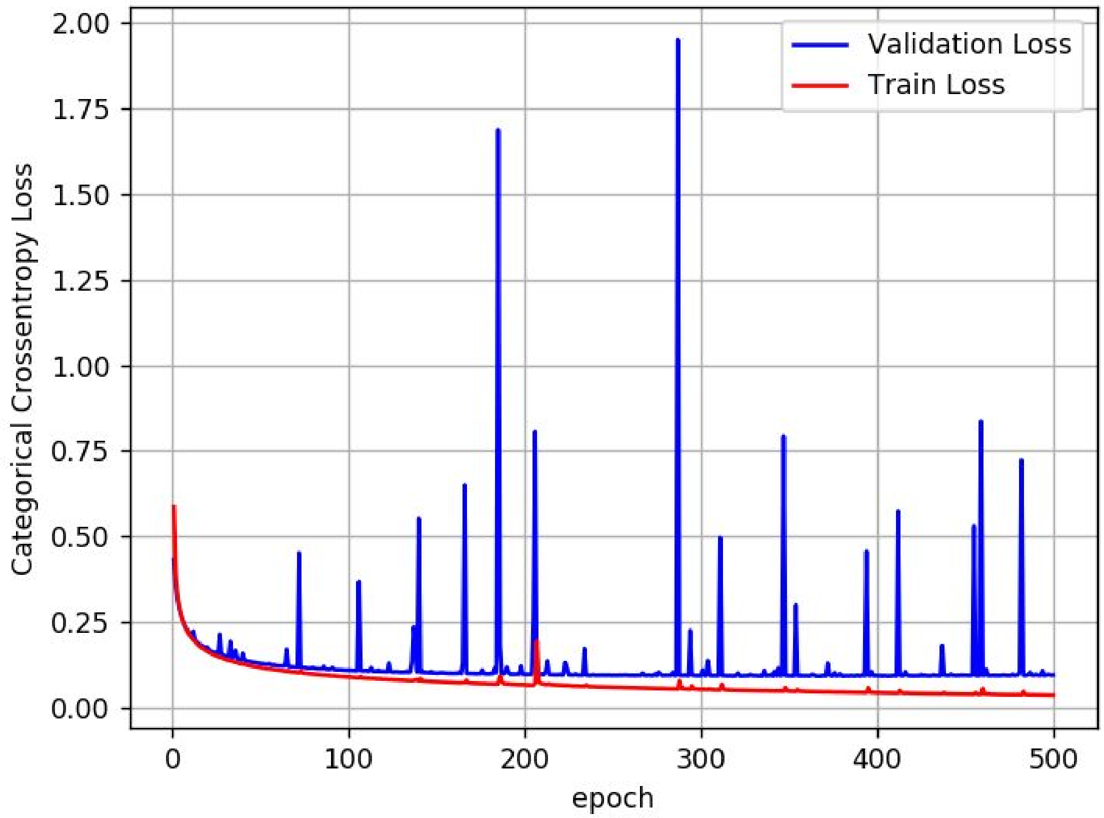
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     18118
           1       0.87      0.70      0.78       556
           2       0.95      0.92      0.93      1448
           3       0.79      0.65      0.71       162
           4       0.98      0.97      0.98      1608

    accuracy                           0.98     21892
   macro avg       0.91      0.85      0.88     21892
weighted avg       0.98      0.98      0.98     21892
```

Test score: 0.09430224171381225

Test accuracy: 0.9774346947669983

Loss Plot



5. **Multi classifier with ReLU and ADAM Optimizer**

In this model we used Rectified linear units (ReLU) as activation function and Adaptive moment estimation as optimizer.

```
Model: "sequential_1"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_1 (Dense)              (None, 128)               24064

_________________________________________________________________

dense_2 (Dense)              (None, 64)                8256

_________________________________________________________________
```

```
dense_3 (Dense)              (None, 5)                 325

=================================================================

Total params: 32,645

Trainable params: 32,645

Non-trainable params: 0

_________________________________________________________________

None
```

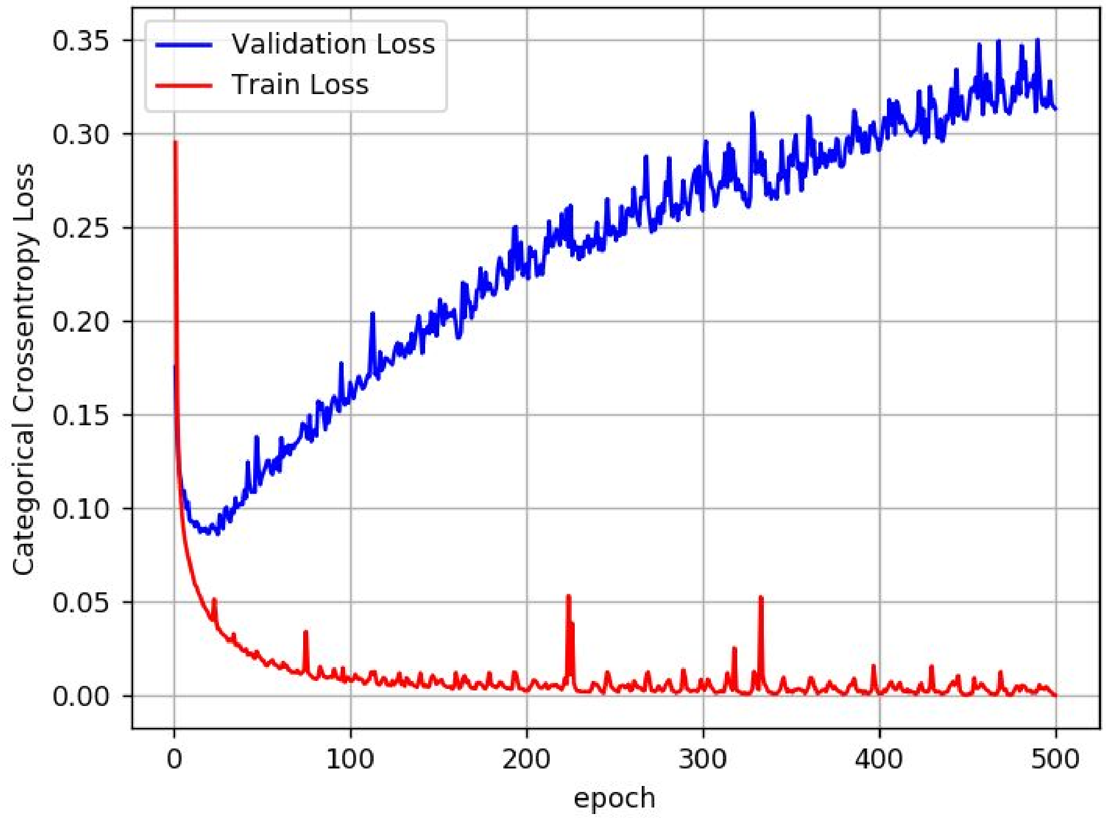
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     18118
           1       0.84      0.75      0.80       556
           2       0.95      0.94      0.94      1448
           3       0.80      0.75      0.77       162
           4       0.99      0.98      0.99      1608

    accuracy                           0.98     21892
   macro avg       0.91      0.88      0.90     21892
weighted avg       0.98      0.98      0.98     21892
```

Test score: 0.31290080993886243

Test accuracy: 0.9801297187805176

Loss Plot



6. Multi classifier with sigmoid, Batch-Norm and Adam Optimizer

In this model we used sigmoid as activation function, Adaptive moment estimation as optimizer (ADAM) and we applied batch normalization on hidden layers.

```
Model: "sequential_1"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_1 (Dense)              (None, 128)               24064

_________________________________________________________________

batch_normalization_1 (Batch (None, 128)               512

_________________________________________________________________
```

```
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
batch_normalization_2 (Batch (None, 64)                256
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 325
=================================================================
Total params: 33,413
Trainable params: 33,029
Non-trainable params: 384
_________________________________________________________________
```

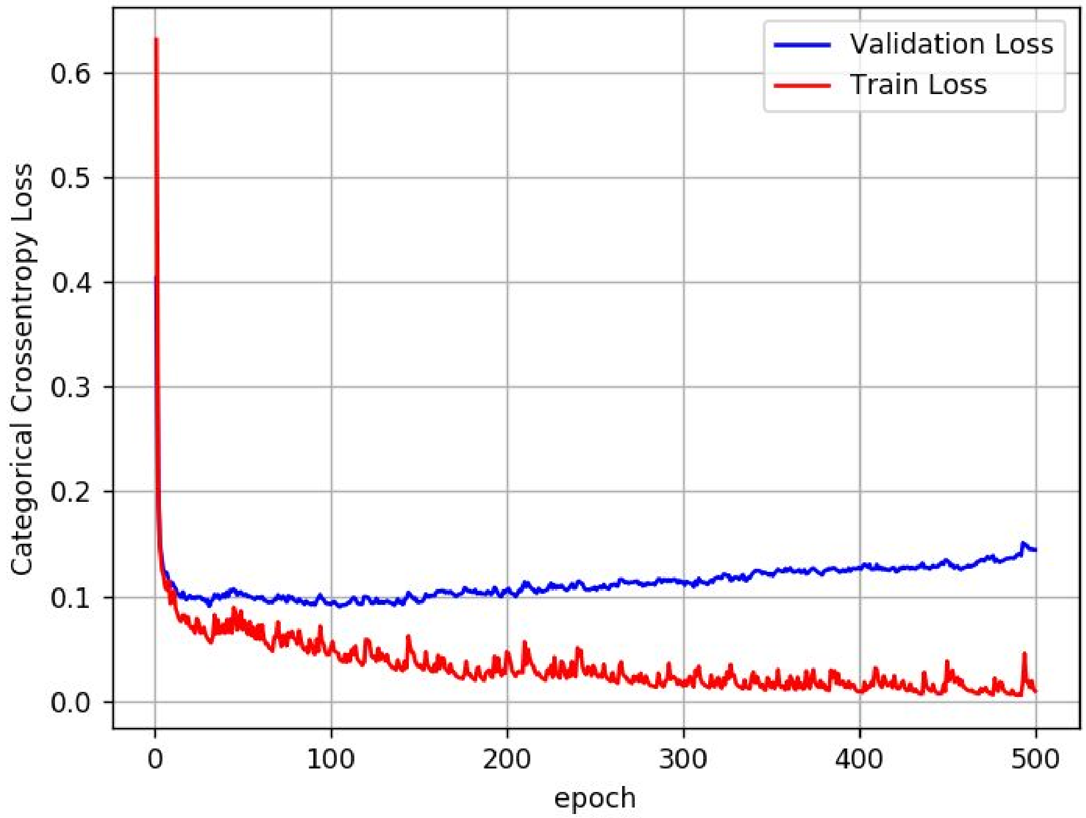
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99     18118
           1       0.83      0.72      0.77       556
           2       0.94      0.93      0.93      1448
           3       0.81      0.71      0.76       162
           4       0.98      0.98      0.98      1608

    accuracy                           0.98     21892
   macro avg       0.91      0.87      0.89     21892
weighted avg       0.98      0.98      0.98     21892
```

Test score: 0.14442630712312768

Test accuracy: 0.9776173830032349

Loss Plot



7. Multi classifier with sigmoid, Dropout and Adam Optimizer

In this model we used sigmoid as activation function and Adaptive moment estimation as optimizer (ADAM) and we also used dropout. A fully connected layer occupies most of the parameters, and hence, neurons develop co-dependency amongst each other during training which curbs the individual power of each neuron leading to over-fitting in training data. That's why we used dropout approach for regularization in neural networks which helps reducing interdependent learning amongst the neurons.

```
Model: "sequential_1"

_________________________________________________________________

Layer (type)                 Output Shape              Param #

=================================================================

dense_1 (Dense)              (None, 128)               24064

_________________________________________________________________
```

```
batch_normalization_1 (Batch (None, 128)              512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)              0
_________________________________________________________________
dense_2 (Dense)              (None, 64)               8256
_________________________________________________________________
batch_normalization_2 (Batch (None, 64)               256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)               0
_________________________________________________________________
dense_3 (Dense)              (None, 5)                325
=================================================================
Total params: 33,413

Trainable params: 33,029

Non-trainable params: 384
_________________________________________________________________
```

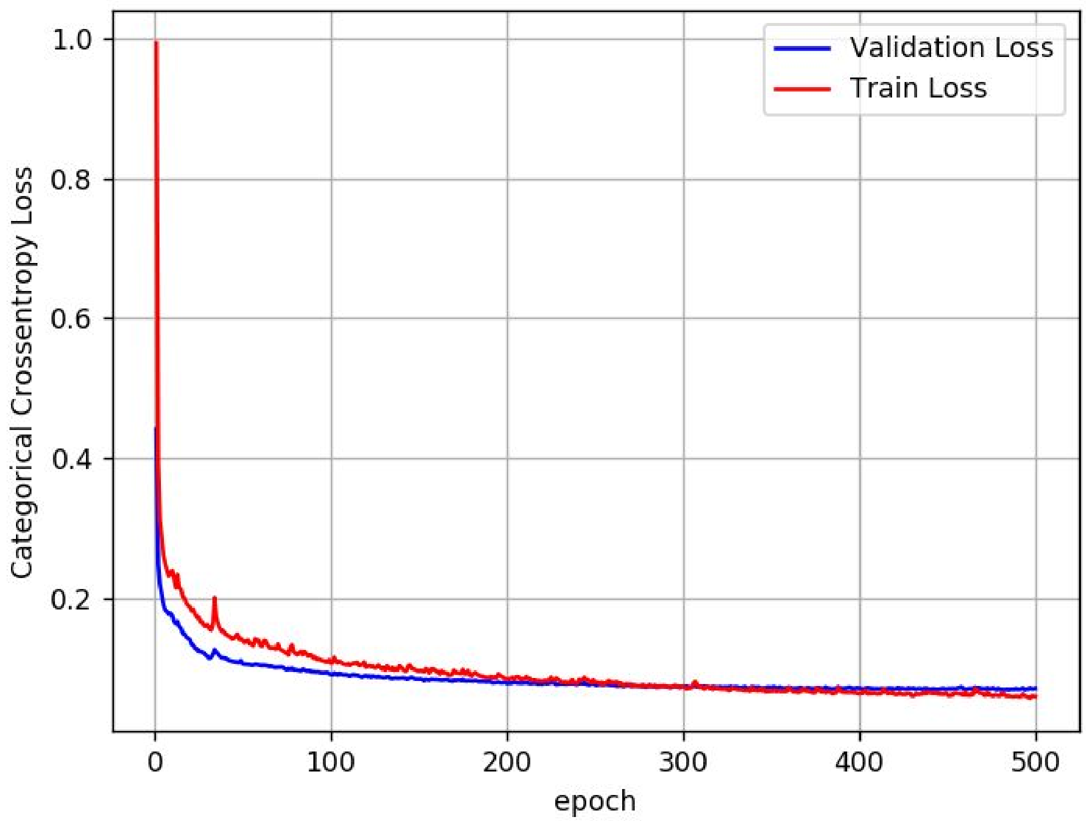
Confusion matrix

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99     18118
           1       0.94      0.68      0.79       556
           2       0.96      0.94      0.95      1448
           3       0.86      0.73      0.79       162
           4       0.99      0.98      0.98      1608

    accuracy                           0.98     21892
   macro avg       0.95      0.87      0.90     21892
weighted avg       0.98      0.98      0.98     21892
```

Test score: 0.07144708678216694

Test accuracy: 0.9815000891685486

Loss Plot



8. **Multi classifier with ReLU, Batch Normalization, Dropout and Adam Optimizer**

In this model we used ReLU as activation function and Adaptive moment estimation as optimizer (ADAM) and we also used batch normalization and dropout. We used Batch normalization to make sure that every batch of the data should be in the normal distribution

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
dense_1 (Dense)              (None, 128)               24064
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
batch_normalization_2 (Batch (None, 64)                256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 325
=================================================================
Total params: 33,413
Trainable params: 33,029
Non-trainable params: 384
_________________________________________________________________
None
```

Confusion matrix

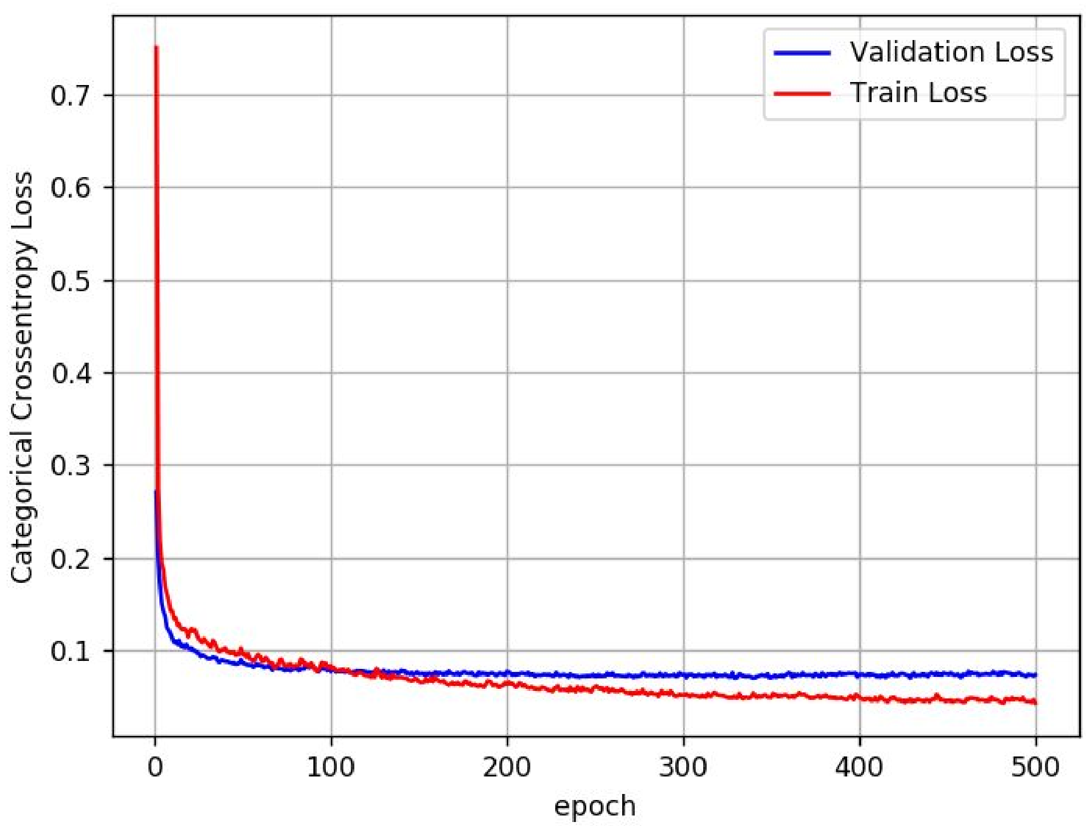
```
        precision    recall  f1-score   support

      0       0.99      1.00      0.99     18118
      1       0.93      0.70      0.80       556
      2       0.96      0.95      0.96      1448
      3       0.90      0.69      0.78       162
      4       0.99      0.98      0.99      1608

accuracy                          0.98     21892
```

macro avg	0.95	0.86	0.90	21892
weighted avg	0.98	0.98	0.98	21892

Test score: 0.07293426106871699

Test accuracy: 0.9828248023986816

Loss Plot



Long short-term memory model multi-class classification:

Below table represents the LSTM model performance.

	precision	recall	f1-score
0	0.99	0.99	0.99
1	0.86	0.73	0.79
2	0.95	0.93	0.94
3	0.80	0.73	0.77
4	0.99	0.97	0.98

Accuracy : 0.98