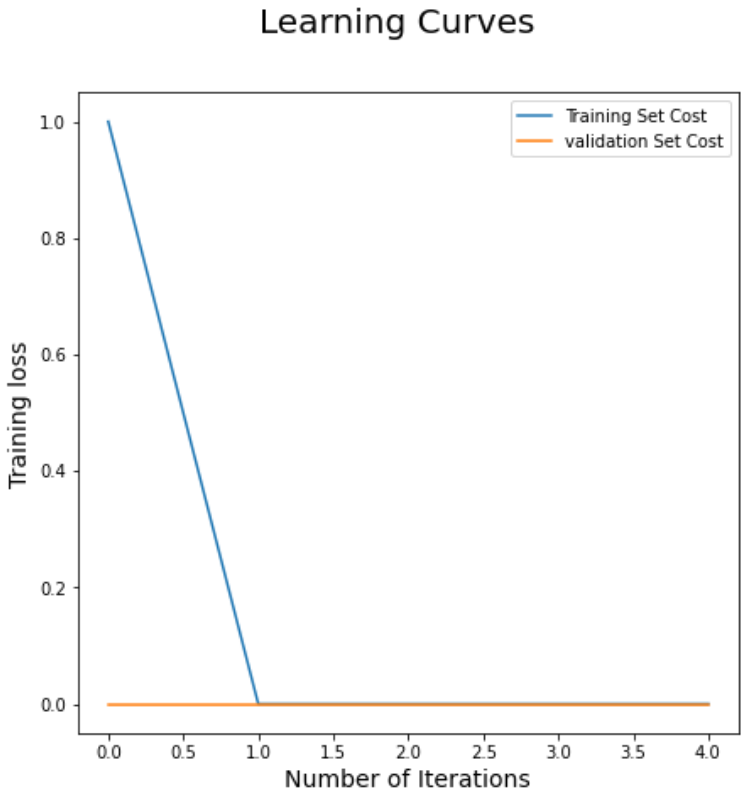
Assignment 2 (Perceptron, Kernel Perceptron, MLP, RBFN, ELM, SVM, Autoencoders, CNN, Fuzzy Network)

Drive Link for all the colab files and datasets-

Assignment-2

Q1) Implement a non-linear perceptron algorithm for the classification using Hebbian Learning rule. The dataset (data55.mat) contains 4 features and the last column is the output (class label). You can use hold-out cross-validation (70, 10, and 20%) for the selection of training, validation and test instances. Evaluate accuracy, sensitivity and specificity measures for the evaluation of test instances. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed).
Colab link = Code
The Loss function variation of training data and validation data are as follows-



The weight vector is obtained as-

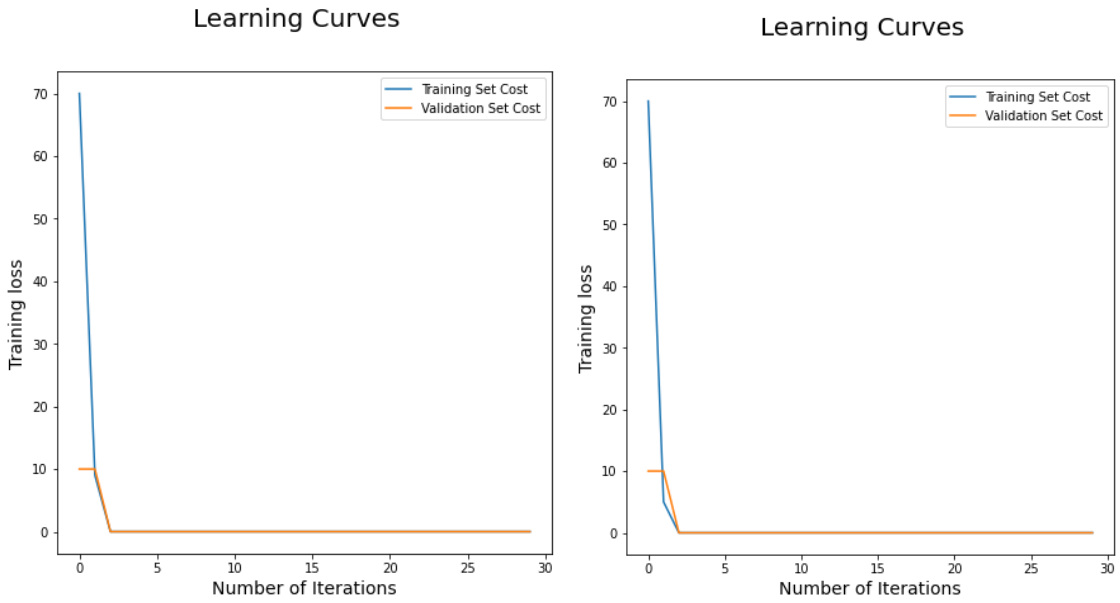
```
[[ 0.034592  ]
 [-0.00948159]
 [ 0.04658916]
 [ 0.06440234]
 [ 0.05      ]]
```

The Accuracy, Sensitivity and Specificity for test data are obtained as 1.0 , 1.0 , 1.0 respectively.

Confusion matrix obtained = [[10 0] [0 10]]

Q2) Implement kernel perceptron algorithm for the classification. The dataset (data55.mat) contains 4 features and the last column is the output (class label). You can use hold-out cross-validation (70, 10, and 20%) for the selection of training, validation and test instances. Evaluate accuracy, sensitivity and specificity measures for the evaluation of test instances. (You can use RBF, and polynomial kernels). (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed).

Colab link = Code
The Loss function variation of training data and validation data are as follows-
For polynomial kernel and rbf kernel respectively,

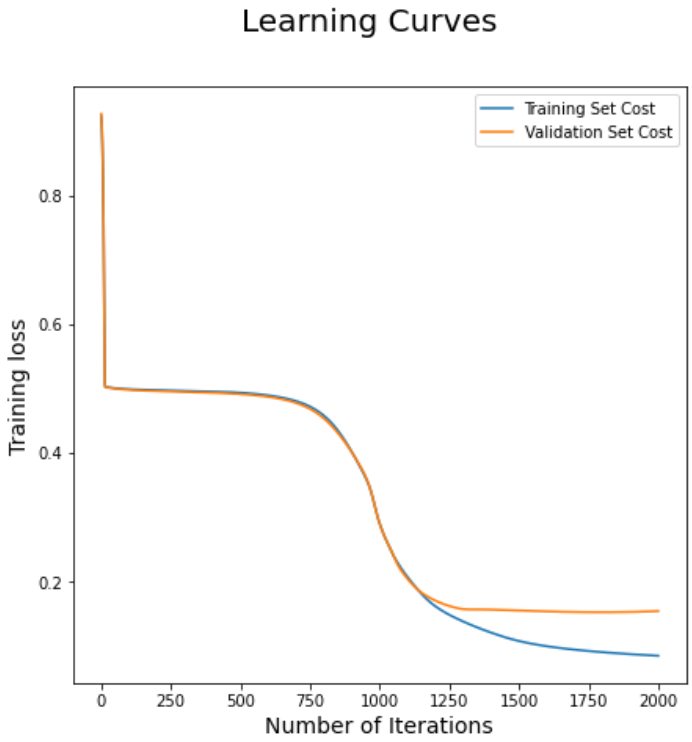


For polynomial kernel,
The Accuracy, Sensitivity and Specificity for test data are obtained as 1.0 , 1.0 , 1.0 respectively.
The Confusion matrix obtained= [[10,0] ; [0,10]]

For RBF kernel,
The Accuracy, Sensitivity and Specificity for test data are obtained as 1.0 , 1.0 , 1.0 respectively.
The Confusion matrix obtained= [[10,0] ; [0,10]]

Q3) The dataset (data5.mat) contains 72 features and the last column is the output (class labels). Design a multilayer perceptron based neural network (two hidden layers) for the classification. You can use both holdout (70, 10, and 20%) and 5-fold cross-validation approaches for evaluating the performance of the classifier (individual accuracy and overall accuracy). You can select the number of hidden neurons of each hidden layer and other MLP parameters using grid-search method. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed).

Colab link = Code
The Loss function variation of training data and validation data in Hold out cross validation are as follows-



In holdout validation,

```
test error = 0.20105308527336394
Training error converged from 0.9243511923086271 to 0.08460947377974891
```

The results obtained from 5-fold on test data are as follows-

```
Accuracies=[0.9930232558139535, 0.9906976744186047, 0.9906976744186047,
0.9953379953379954, 0.9953379953379954]

Sensitivity=[0.9952380952380953, 0.9863636363636363, 0.9951219512195122,
1.0, 0.9908256880733946]

Specificity=[0.990909090909091, 0.9952380952380953, 0.9866666666666667,
0.9903381642512077, 1.0]
```

Confusion matrices WRT 5 folds are obtained as-

```
[[209   2]
 [  1 218]]

[[217   1]
 [  3 209]]
[[204   3]
 [  1 222]]
[[222   2]
 [  0 205]]

[[216   0]
 [  2 211]]

Training error converged from 0.9442251156250331 to 0.010370703841280434

Validation error converged from 0.9464596489775168 to 0.07593971207258458

Test error converged from 0.014125692622248578 to 0.009341228885850133
```

Q4) Implement the radial basis function neural network (RBFNN) for the classification problem. You can use Gaussian, multiquadric and linear kernel functions for the implementation. You can use both holdout (70, 10, and 20%) and 5-fold cross-validation approaches for evaluating the performance of the classifier (individual accuracy and overall accuracy). The dataset (data5.mat) contains 72 features and the last column is the output (class labels). (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed).

Colab link = Code

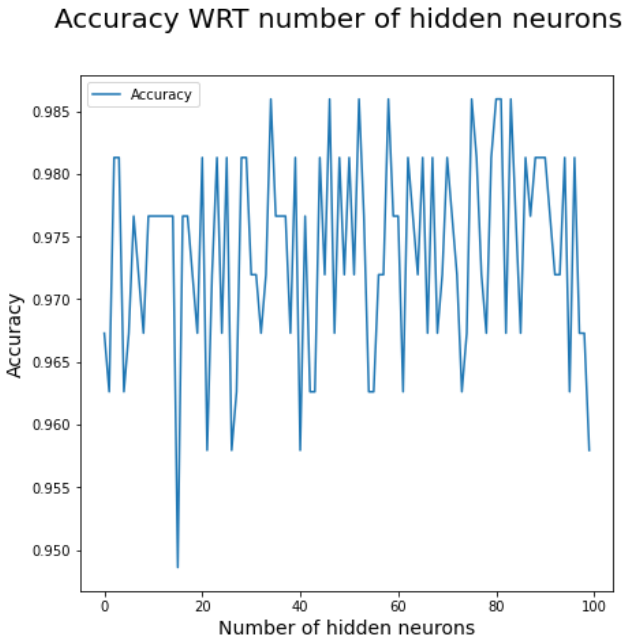
On using Linear kernel -

In holdout validation the test classification,

```
Accuracies=[0.9488372093023256]

Sensitivity=[0.9162790697674419]

Specificity=[0.9813953488372092]
```

Confusion matrix- **[** `[197,   4]`
`[ 18, 211] ]`

The Grid search result obtained for number of hidden neurons is - 480



The results obtained from 5-fold on test data are as follows-

```
Accuracies=[0.9767441860465116, 0.9767441860465116, 0.9883720930232558,
0.9825581395348837, 0.9825581395348837]

Sensitivity=[0.9770114942528736, 0.9753086419753086, 0.9883720930232558,
0.9655172413793104, 0.9666666666666667]

Specificity=[0.9764705882352941, 0.978021978021978, 0.9883720930232558,
1.0, 1.0]
```

Confusion matrices WRT 5 folds are obtained as-

```
[[85  2]
 [ 2 83]]

[[79  2]
 [ 2 89]]

[[85  1]
 [ 1 85]]

[[84  0]
 [ 3 85]]

[[87  0]
 [ 3 82]]
```

On using gaussian kernel -

In holdout validation the test classification,

```
Accuracies=[0.9511627906976744]


Sensitivity=[0.958139534883721]

Specificity=[0.9441860465116279]

[[206,  12],
 [  9, 203] ]
```

The results obtained from 5-fold on test data are as follows-

```
Accuracies=[0.9883720930232558, 0.9941860465116279, 0.9883720930232558,
0.9883720930232558, 0.9767441860465116]

Sensitivity=[0.9885057471264368, 0.9891304347826086, 0.9753086419753086,
1.0, 0.9555555555555556]

Specificity=[0.9882352941176471, 1.0, 1.0, 0.9782608695652174, 1.0]

[[86  1]
 [ 1 84]]

[[91  0]
 [ 1 80]]

[[79  0]
 [ 2 91]]

[[80  2]
 [ 0 90]]

[[86  0]
 [ 4 82]]
```

On using multiquadric kernel -

In holdout validation the test classification,

```
Accuracies=[0.9558139534883721]

Sensitivity=[0.9627906976744186]

Specificity=[0.9488372093023256]

[ [207,  11],
  [  8, 204] ]
```

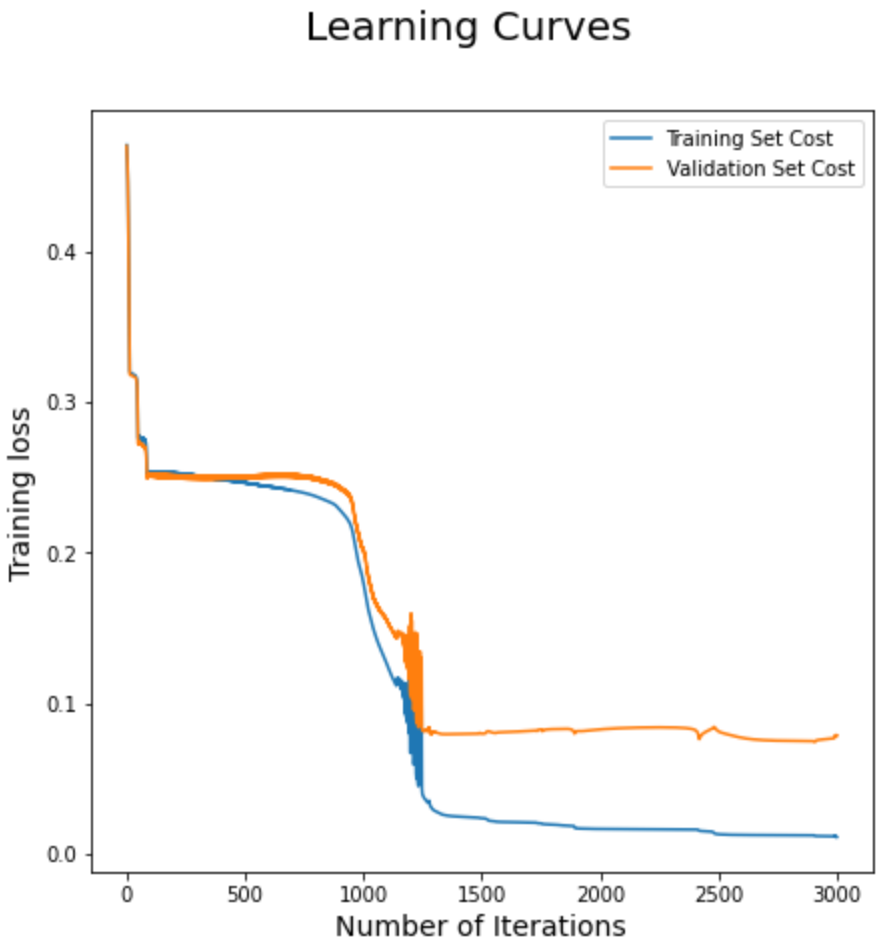
The results obtained from 5-fold on test data are as follows-

```
Accuracies=[0.9883720930232558, 0.9825581395348837, 0.9883720930232558,
0.9941860465116279, 0.9651162790697675]

Sensitivity=[0.9885057471264368, 0.967391304347826, 0.9876543209876543,
1.0, 0.9666666666666667]

Specificity=[0.9882352941176471, 1.0, 0.989010989010989,
0.9891304347826086, 0.9634146341463414]

[[86  1]
 [ 1 84]]

[[89  0]
 [ 3 80]]

[[80  1]
 [ 1 90]]

[[80  1]
 [ 0 91]]

[[87  3]
 [ 3 79]]
```

Q5) Implement the stacked autoencoder based deep neural network for the classification problem. The deep neural network must contain 3 hidden layers from three autoencoders. You can use holdout (70, 10, and 20%) cross-validation technique for selecting, training and test instances for the classifier. The dataset (data5.mat) contains 72 features and the last column is the output (class labels). For autoencoder implementation, please use back propagation algorithm which has been already taught in the class. Evaluate individual accuracy and overall accuracy. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed)

Colab link = Code

The Loss function variation of training data and validation data in Hold out cross validation are as follows-



The final test error obtained (`Test error`)=0.08055459755409089

`The outputs obtained are-`

```
Train error converged from 0.47028799115096465 to 0.01138991116876878

Validation error converged from 0.4696306737716704 to 0.07884129190139882
```

```
Training accuracy and Testing accuracy are [0.9886892880904857,
0.9069767441860465] respectively
```

In the test classification,

```
Accuracy=0.9069767441860465

Sensitivity=0.9023255813953488

Specificity=0.9116279069767442

The confusion matrix obtained on the test data-
[[194  19]
 [ 21 196]]
```

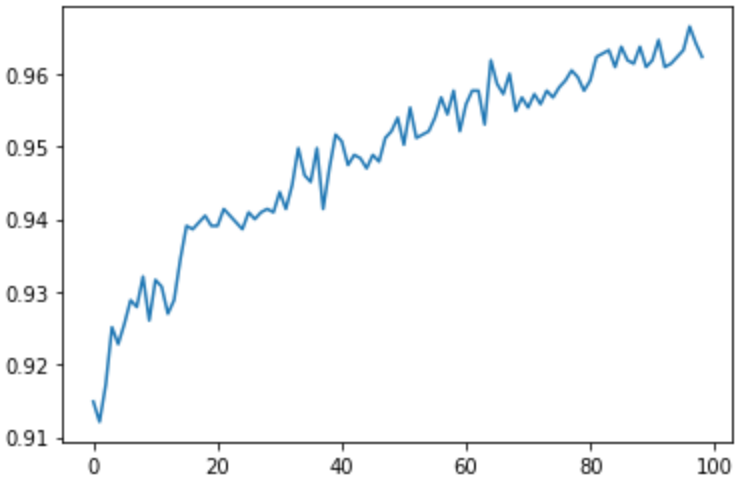
Q6) Implement an extreme learning machine (ELM) classifier for the classification. You can use Gaussian and tanh activation functions. Please select the training and test instances using 5-fold cross- validation technique Evaluate individual accuracy and overall accuracy. The dataset (data5.mat) contains 72 features and the last column is the output (class labels). (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed).

Colab link = Code

The Outputs obtained with tanh activation are-

```
Accuracies=[0.9720930232558139, 0.9837209302325581, 0.9790697674418605,
0.9790209790209791, 0.972027972027972]

Sensitivity=[0.9955555555555555, 0.9681818181818181, 0.958139534883721,
0.958139534883721, 0.945]

Specificity=[0.9463414634146341, 1.0, 1.0, 1.0, 0.9956331877729258]

Confusion matrices for 5 folds-
[[224  11]
 [  1 194]]

[[213   0]
 [  7 210]]

[[206   0]
 [  9 215]]

[[206   0]
 [  9 214]]

[[189   1]
 [ 11 228]]
```

On performing Gridsearch for ideal number of hidden neurons -



It is clear that accuracy increases with the number of hidden neurons.

The Outputs obtained with Gaussian activation are-

```
Accuracies=[0.9395348837209302, 0.9116279069767442, 0.9279069767441861,
0.9230769230769231, 0.9207459207459208]

Sensitivity=[0.8706467661691543, 1.0, 1.0, 1.0, 0.8425925925925926]

Specificity=[1.0, 0.8240740740740741, 0.845771144278607,
0.8457943925233645, 1.0]

[[175   0]
 [ 26 229]]

[[214  38]
 [  0 178]]

[[229  31]
 [  0 170]]

[[215  33]
 [  0 181]]

[[182   0]
 [ 34 213]]
```

Q7) Implement a deep neural network, which contains two hidden layers (the hidden layers are obtained from the ELM-autoencoders). The last layer will be the ELM layer which means the second hidden layer feature vector is used as input to the ELM classifier. The network can be called as deep layer stacked autoencoder based extreme learning machine. You can use holdout approach (70, 10, 20%) for evaluating the performance of the classifier. The dataset (data5.mat) contains 72 features and the last column is the output (class labels). Evaluate individual accuracy and overall accuracy. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed)

Colab link = Code
The outputs are as follows-

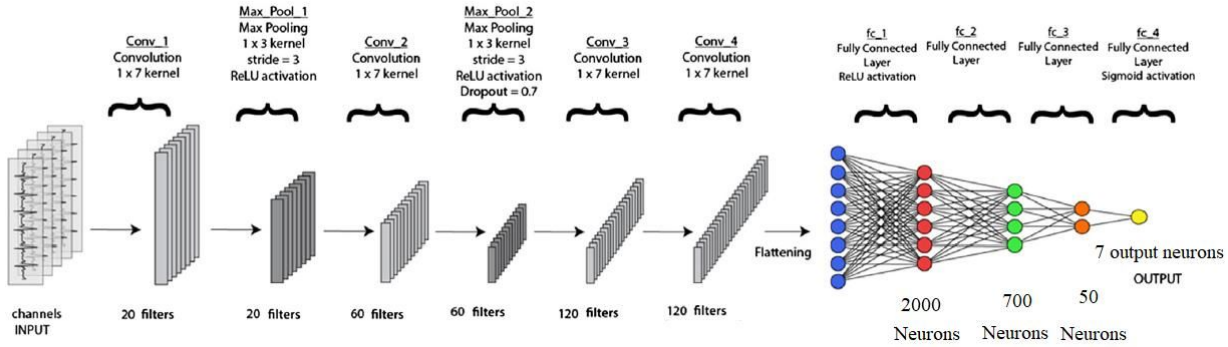
```
Accuracy=0.7837209302325582

Sensitivity=0.7953488372093023

Specificity=0.772093023255814

[[171  49]
 [ 44 166]]
```

Q8) Implement support vector machine (SVM) classifier for the multi-class classification task. You can use one vs one and one vs all multiclass coding methods to create binary SVM models. Implement the SMO algorithm for the evaluation of the training parameters of SVM such as Lagrange multipliers. You can use a holdout approach (70, 10, 20%) for evaluating the performance of the classifier. The dataset (data5.mat) contains 72 features and the last column is the output (class labels). Evaluate individual accuracy and overall accuracy. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are not allowed)

Colab link = Code
The Output obtained from the SVM classifier is =

```
Accuracy=0.873953488372093

Sensitivity=0.9377777777777778

Specificity=0.8048780487804878

[[211  66]
 [ 14 139]]
```

Q9) Implement a multi-channel 1D deep CNN architecture for the seven-class classification task. The input and the class labels are given in .mat file format. There is a total of 17160 number of instances present in both input and class-label data files. The input data for each instance is a multichannel time series (12-channel) with size as (12 ×800). The class label for each multichannel time series instance is given in the class_label.mat file. You can select the training and test instances using hold- out cross-validation (70% training, 10% validation, and 20% testing). The architecture of the multi- channel deep CNN is given as follows. The number of filters, length of each filter, and number of neurons in the fully connected layers are shown in the following figure. Evaluate individual accuracy and overall accuracy. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are allowed)



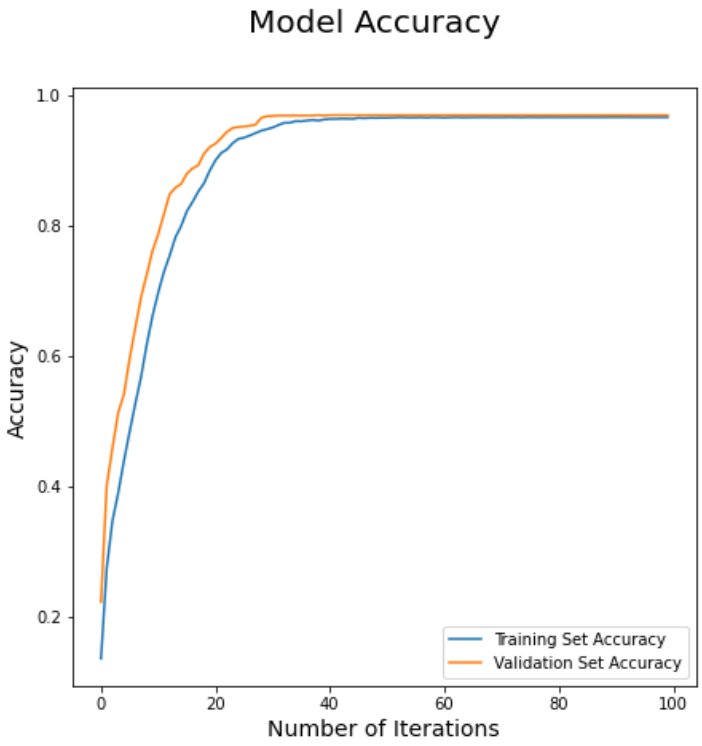
Colab link = [Code]

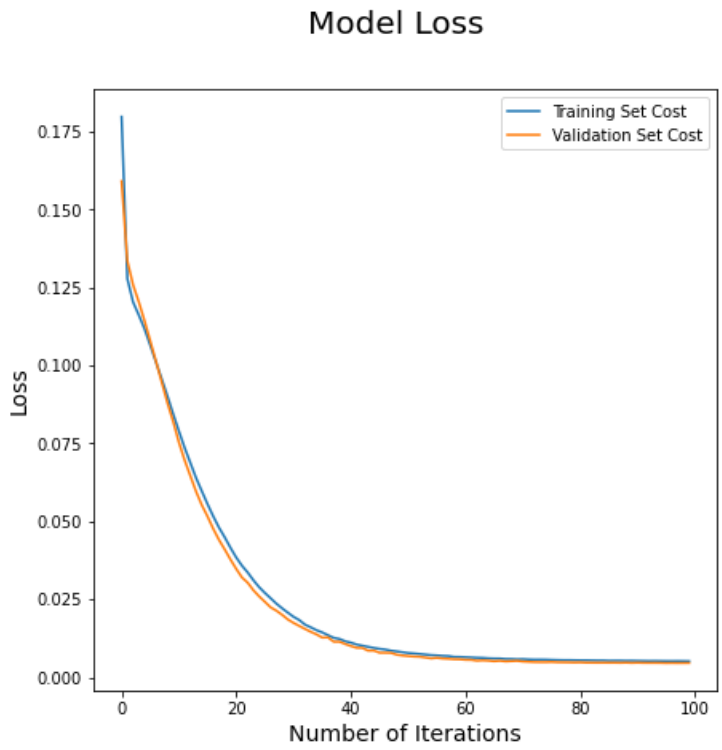
The final test accuracy is obtained as = `0.992132842540741`

Layer (type)	Output Shape	Param #
conv1d_3 (Conv1D)	(None, 800, 20)	1700
max_pooling1d_2 (MaxPooling1	(None, 267, 20)	0
conv1d_4 (Conv1D)	(None, 267, 60)	8460
max_pooling1d_3 (MaxPooling1	(None, 89, 60)	0
dropout_1 (Dropout)	(None, 89, 60)	0
conv1d_5 (Conv1D)	(None, 89, 120)	50520
flatten_1 (Flatten)	(None, 10680)	0
dense_4 (Dense)	(None, 2000)	21362000
dense_5 (Dense)	(None, 700)	1400700
dense_6 (Dense)	(None, 50)	35050
dense_7 (Dense)	(None, 7)	357

Total params: 22,858,787
Trainable params: 22,858,787
Non-trainable params: 0

The trend in which the Accuracy and Loss changed is -





Q10) The two-dimensional time-frequency images of class1, class2, class3 are given in the folders as class1.zip, class2.zip, class3.zip, respectively. Design a 2D deep CNN classifier for the three-class classification. Evaluate the classification performance using hold-out cross-validation (70% training, 10% validation, 20% testing), and 10-fold cross-validation methods. Evaluate individual accuracy and overall accuracy for the multiclass CNN classifier. You can consider 4 convolutional layer, three pooling layer, and 5 fully connected layers. You can select the number of filters, stride for convolution and pooling layers, and number of neurons for fully connected layers as per your own choice. (Packages such as keras, tensorflow, pytorch for python and MATLAB deep learning toolbox etc. are allowed). You can apply dropout, batch normalization, and regularization to improve the classification performance.

Colab link = Code

The architecture used -

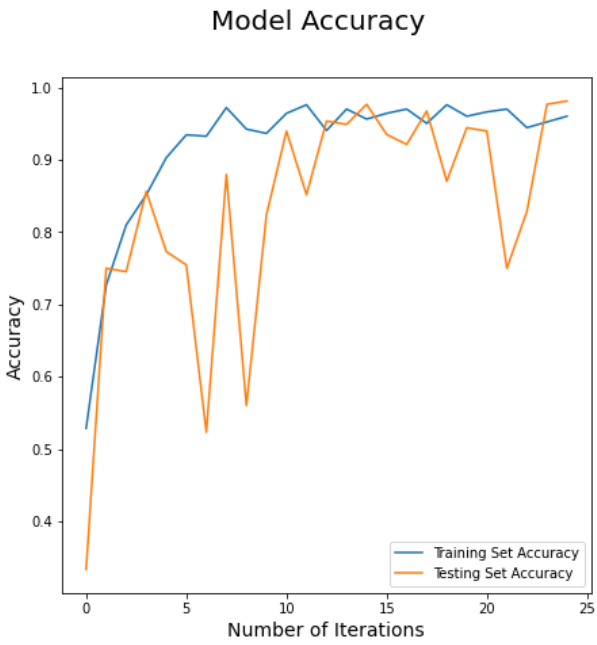
Layer (type)	Output Shape	Param #
conv2d_73 (Conv2D)	(None, 656, 875, 10)	280
batch_normalization_98 (Batc	(None, 656, 875, 10)	40
max_pooling2d_45 (MaxPooling	(None, 328, 438, 10)	0
conv2d_74 (Conv2D)	(None, 164, 219, 20)	1820
batch_normalization_99 (Batc	(None, 164, 219, 20)	80
max_pooling2d_46 (MaxPooling	(None, 82, 110, 20)	0
dropout_45 (Dropout)	(None, 82, 110, 20)	0
conv2d_75 (Conv2D)	(None, 41, 55, 40)	7240
batch_normalization_100 (Bat	(None, 41, 55, 40)	160
max_pooling2d_47 (MaxPooling	(None, 21, 28, 40)	0
conv2d_76 (Conv2D)	(None, 11, 14, 60)	21660
batch_normalization_101 (Bat	(None, 11, 14, 60)	240
flatten_13 (Flatten)	(None, 9240)	0
dense_56 (Dense)	(None, 2000)	18482000
batch_normalization_102 (Bat	(None, 2000)	8000
dense_57 (Dense)	(None, 500)	1000500
batch_normalization_103 (Bat	(None, 500)	2000
dense_58 (Dense)	(None, 200)	100200
batch_normalization_104 (Bat	(None, 200)	800
dense_59 (Dense)	(None, 20)	4020
batch_normalization_105 (Bat	(None, 20)	80
dense_60 (Dense)	(None, 3)	63
batch_normalization_106 (Bat	(None, 3)	12
dense_61 (Dense)	(None, 3)	12

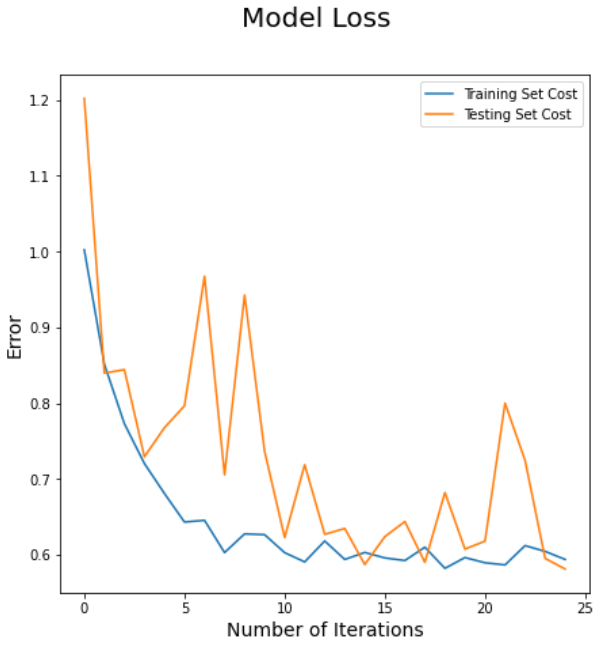
Total params: 19,629,207
Trainable params: 19,623,501
Non-trainable params: 5,706

The Outputs obtained from holdout validation -

The final test accuracy is obtained as = `0.9814814925193787`

The trend in which the Accuracy and Loss changed is -





The Outputs obtained from 10-Fold cross validation -

```
-------------------------------------------------
fold 1 Test Accuracy = 0.9452054500579834
fold 1 Train Accuracy = 0.9691358208656311
-------------------------------------------------
fold 2 Test Accuracy = 1.0
fold 2 Train Accuracy = 0.9799692034721375
-------------------------------------------------
fold 3 Test Accuracy = 0.9722222089767456
fold 3 Train Accuracy = 0.9630200266838074
-------------------------------------------------
fold 4 Test Accuracy = 1.0
fold 4 Train Accuracy = 0.9784283638000488
-------------------------------------------------
fold 5 Test Accuracy = 0.9444444179534912
fold 5 Train Accuracy = 0.990755021572113
-------------------------------------------------
fold 6 Test Accuracy = 0.9722222089767456
fold 6 Train Accuracy = 0.9784283638000488
-------------------------------------------------
fold 7 Test Accuracy = 0.9166666865348816
fold 7 Train Accuracy = 0.9799692034721375
```

```
------------------------------------------------
fold 8 Test Accuracy = 0.9694444179534912
fold 8 Train Accuracy = 0.9830508232116699
------------------------------------------------
fold 9 Test Accuracy = 0.9861111044883728
fold 9 Train Accuracy = 0.9815100431442261
------------------------------------------------
fold 10 Test Accuracy = 0.9611111044883728
fold 10 Train Accuracy = 0.964560866355896
------------------------------------------------
Best model has 100% Test Accuracy

The Overall performance of the model = 0.9667427599430084
```