E1 213: Pattern Recognition and Neural Networks **Spring 2021**

Assignment 3: Non-Linear Models

Author: Abijith J. Kamath *Email:* abijithj@iisc.ac.in

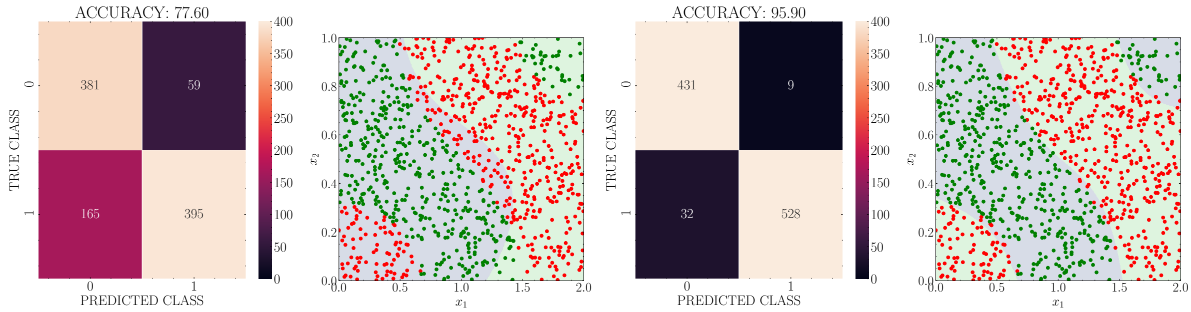
1 SVM for Classification on 2D Synthetic Data

The classification task is to classify 2D features in two classes. Support vector machines (SVM) with polynomial and Gaussian kernel are used for classification along with logistic regression for comparison.

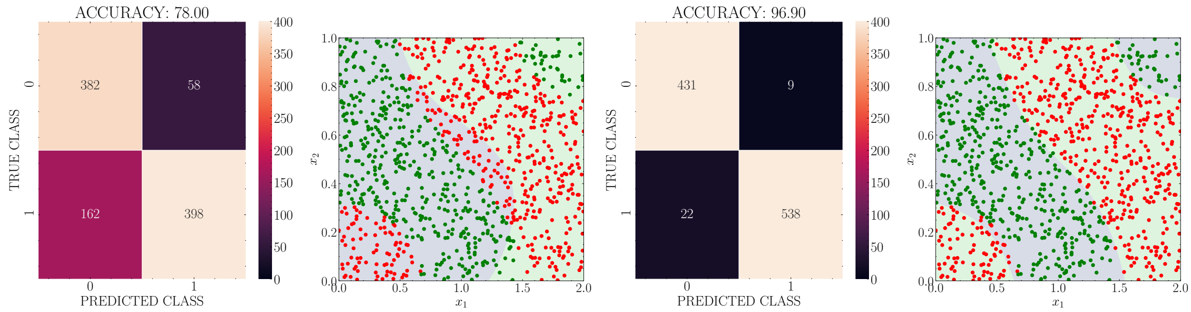
Implementation: The C-SVM with polynomial and Gaussian kernels is trained with varying training sizes using the SVM class from scikit-learn. The parameter $C = 1$ and the degree of the polynomial kernel is set to 3, for all experiments. The classifiers are trained with training sizes varying from 500, 1000, 1500, 2000 and the classifiers are tested on remaining unseen samples.

Results: Figure 1, 3 and 5 shows the shows the accuracies of classification in confusion matrices and the samples along with the discriminant function for classification using SVMs. The first and third column shows the accuracies in confusion matrices with varying training sizes for SVM classifier with polynomial and Gaussian kernels respectively. The second and fourth column shows the corresponding samples with the discriminant functions. Figures 2, 4 and 6 shows the accuracies of classification in confusion matrices and the samples along with the discriminant function for classification using logistic regression. The first row shows the accuracies in confusion matrices with varying training size and the second row shows the corresponding samples with the discriminant functions.

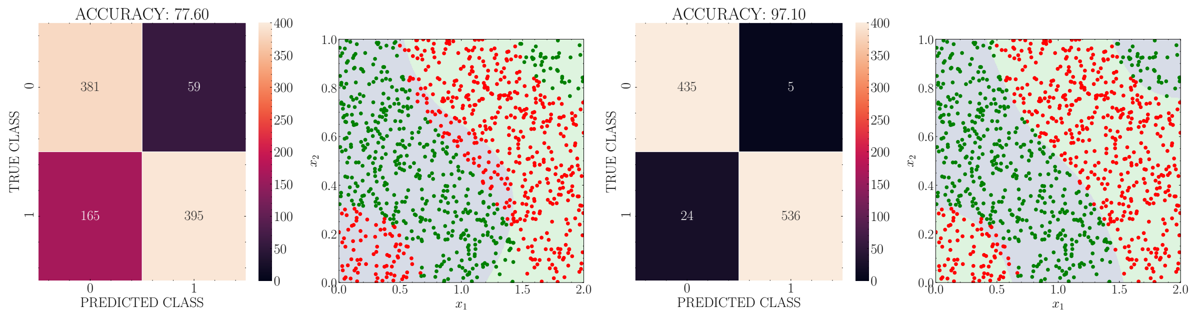
Inferences: It can be seen that the accuracy increases with increasing training size from 500, 100, 1500, and 2000 in both classifiers. Between the classifiers, SVM with RBF kernel outperforms SVM with polynomial kernel. This can also be seen in the samples where the RBF kernel allows learning multimodal distributions within the region, where as the polynomial kernel is restricted by its degree to give smooth discriminative functions.



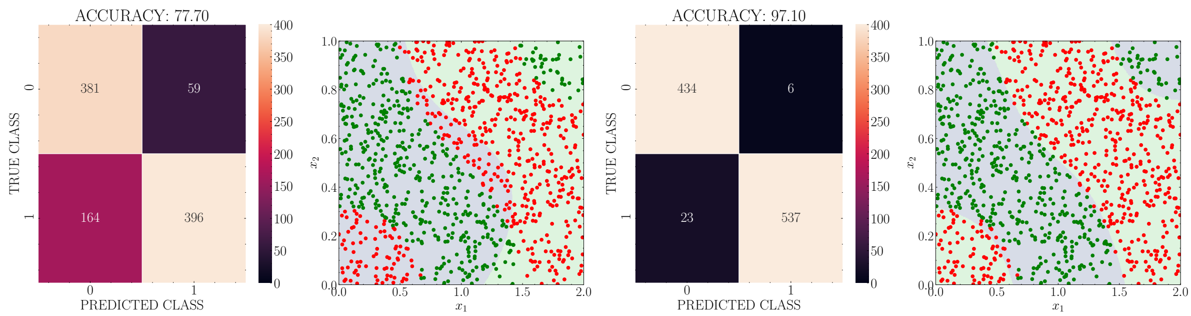
(a) Poly, Size 500 (b) SVM-Poly, Classifier (c) RBF, Size 500 (d) SVM-RBF Classifier



(e) Poly, Size 1000 (f) SVM-Poly, Classifier (g) RBF, Size 1000 (h) SVM-RBF Classifier

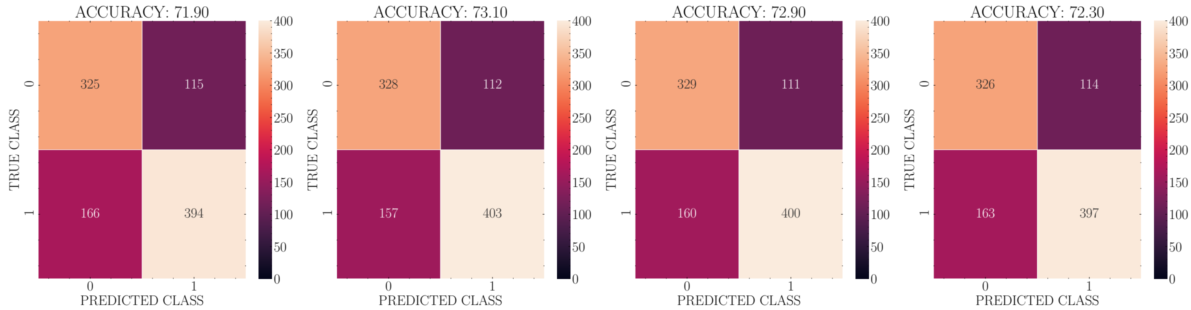


(i) Poly, Size 1500 (j) SVM-Poly, Classifier (k) RBF, Size 1500 (l) SVM-RBF Classifier

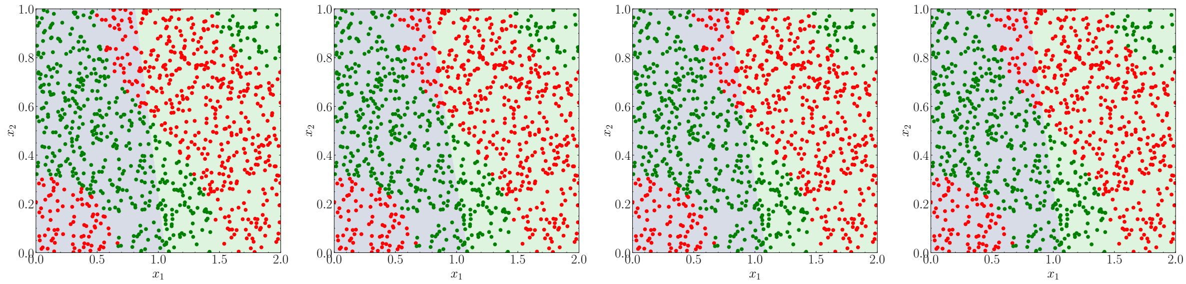


(m) Poly, Size 2000 (n) SVM-Poly, Classifier (o) RBF, Size 2000 (p) SVM-RBF Classifier

Figure 1: Support vector machines with polynomial kernel of degree 3 and Gaussian kernel on 2D data with no label noise.



(a) Poly, Size 500 (b) Poly, Size 1000 (c) Poly, Size 1500 (d) Poly, Size 2000

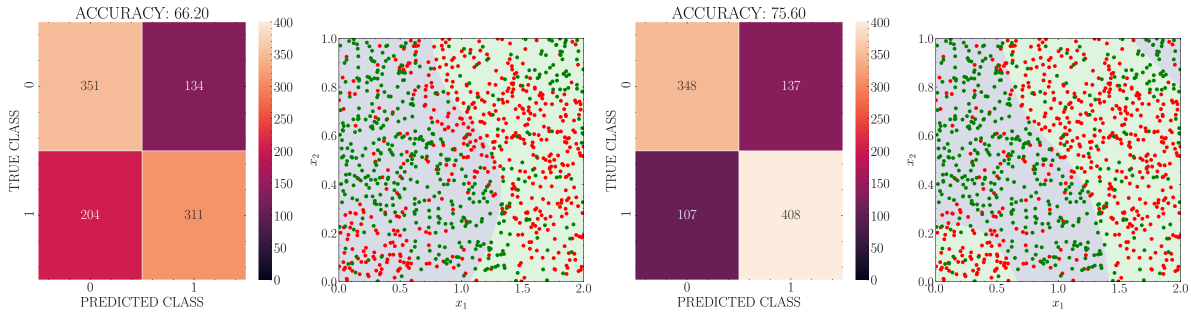


(e) LOG, Classifier (f) LOG, Classifier (g) LOG, Classifier (h) LOG, Classifier

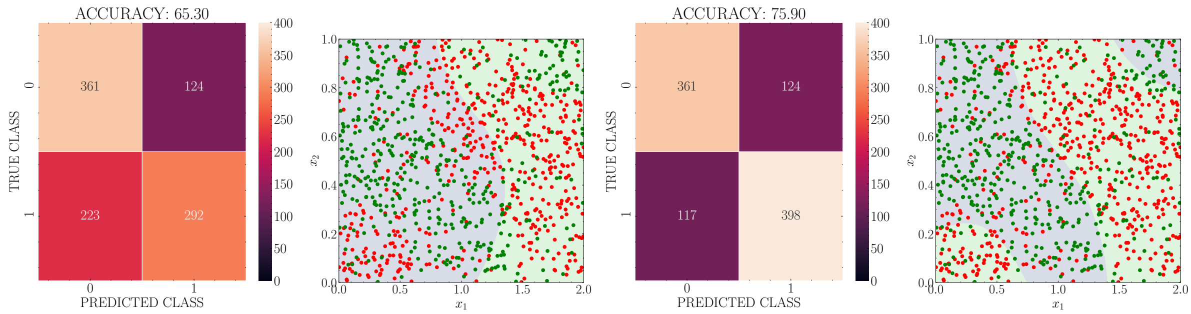
Figure 2: Logistic regression on 2D data with no label noise.

Logistic regression performs with accuracies around 70% which is far less than SVM with RBF kernel. This is to be expected as the logistic regression defines a linear classifier, but the data is not linearly separable.

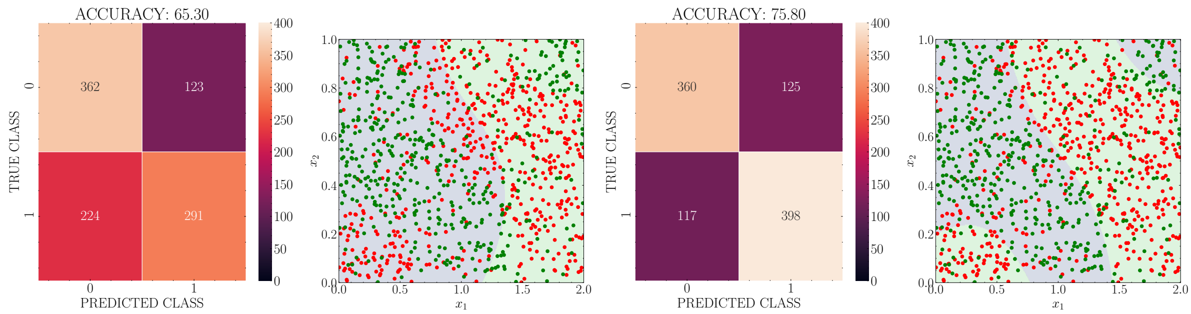
With 20% and 40% label noise, the accuracies of the classifiers decrease. SVM with RBF kernel performs the best, followed by polynomial kernel and then logistic regression.



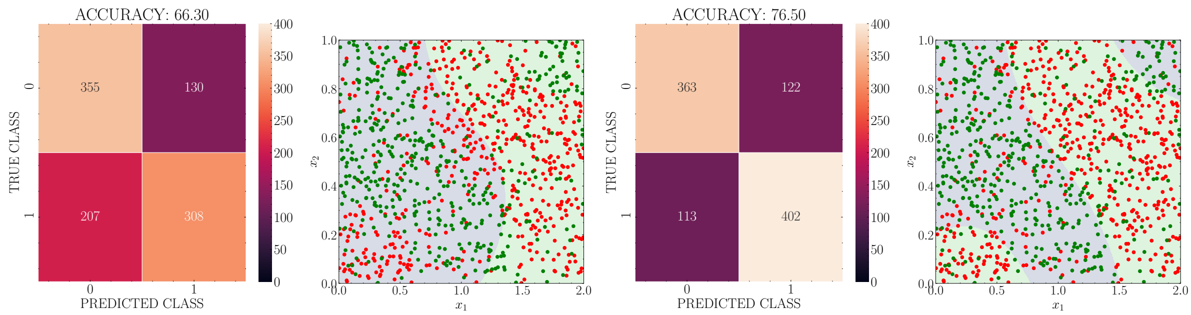
(a) Poly, Size 500 (b) SVM-Poly, Classifier (c) RBF, Size 500 (d) SVM-RBF Classifier



(e) Poly, Size 1000 (f) SVM-Poly, Classifier (g) RBF, Size 1000 (h) SVM-RBF Classifier

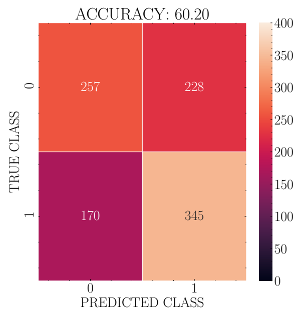


(i) Poly, Size 1500 (j) SVM-Poly, Classifier (k) RBF, Size 1500 (l) SVM-RBF Classifier

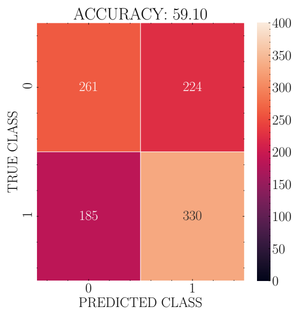


(m) Poly, Size 2000 (n) SVM-Poly, Classifier (o) RBF, Size 2000 (p) SVM-RBF Classifier

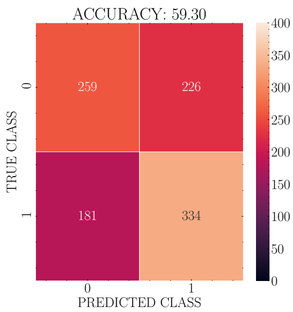
Figure 3: Support vector machines with polynomial kernel of degree 3 and Gaussian kernel on 2D data with 20% label noise.



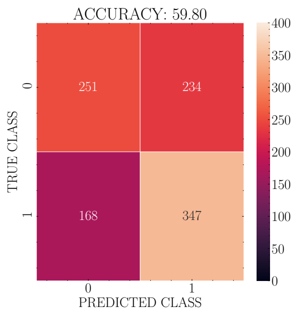
(a) Poly, Size 500



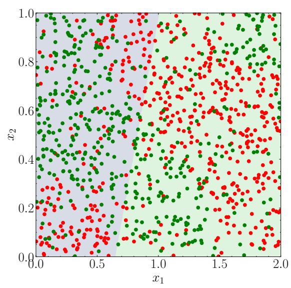
(b) Poly, Size 1000



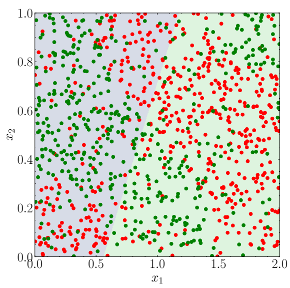
(c) Poly, Size 1500



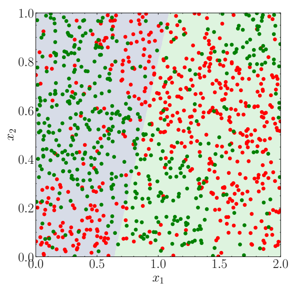
(d) Poly, Size 2000



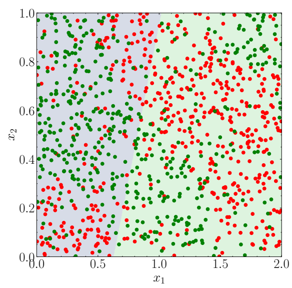
(e) LOG, Classifier



(f) LOG, Classifier

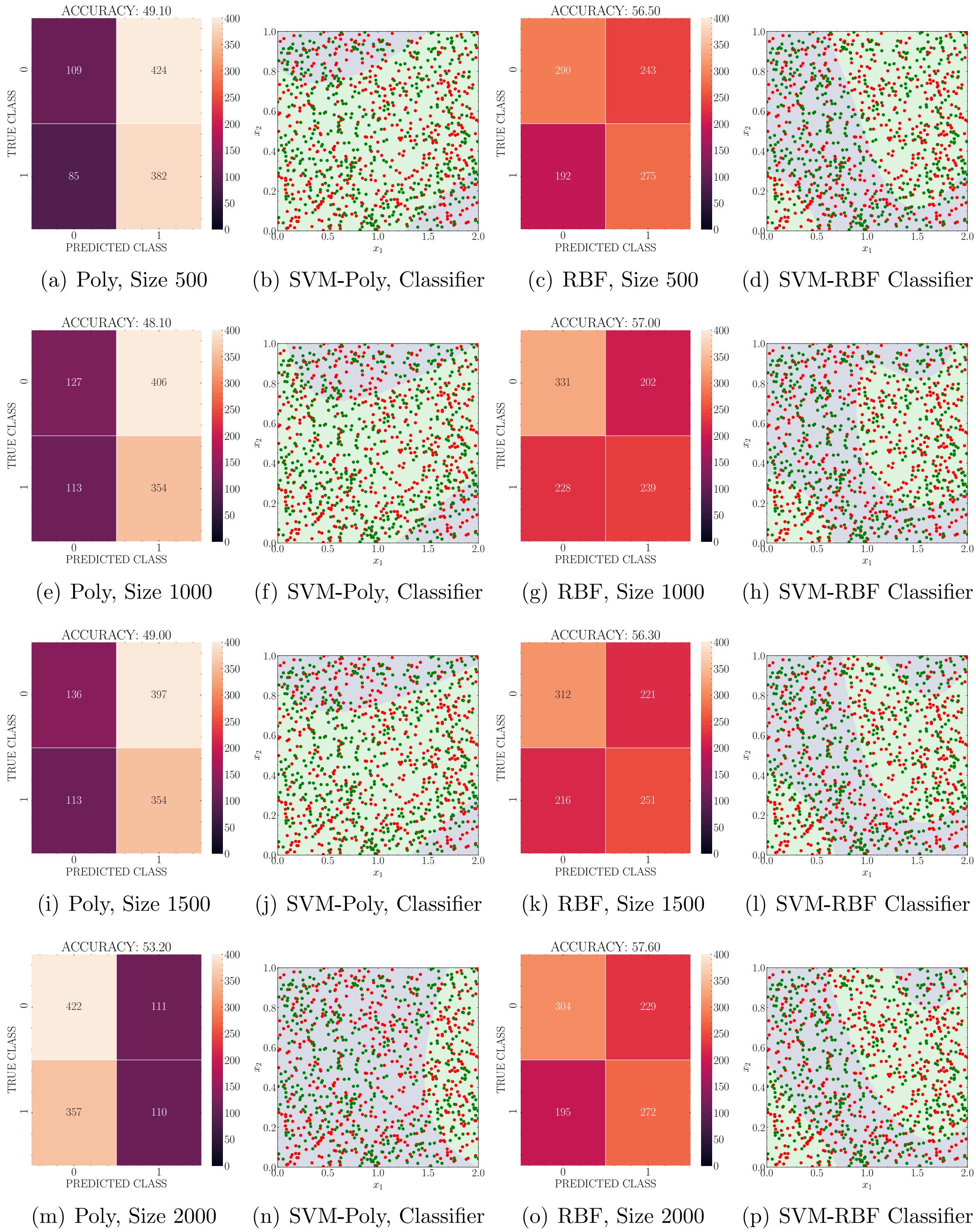


(g) LOG, Classifier



(h) LOG, Classifier

Figure 4: Logistic regression on 2D data with 20% label noise.



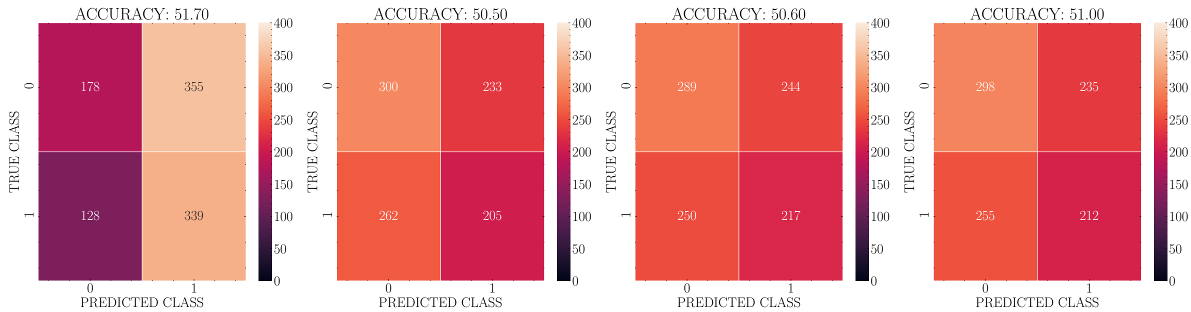
(a) Poly, Size 500 (b) SVM-Poly, Classifier (c) RBF, Size 500 (d) SVM-RBF Classifier

(e) Poly, Size 1000 (f) SVM-Poly, Classifier (g) RBF, Size 1000 (h) SVM-RBF Classifier

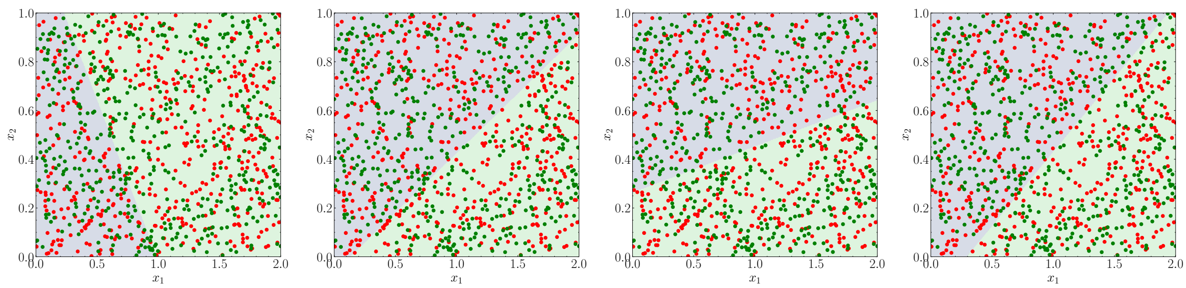
(i) Poly, Size 1500 (j) SVM-Poly, Classifier (k) RBF, Size 1500 (l) SVM-RBF Classifier

(m) Poly, Size 2000 (n) SVM-Poly, Classifier (o) RBF, Size 2000 (p) SVM-RBF Classifier

Figure 5: Support vector machines with polynomial kernel of degree 3 and Gaussian kernel on 2D data with 40% label noise.



(a) Poly, Size 500 (b) Poly, Size 1000 (c) Poly, Size 1500 (d) Poly, Size 2000



(e) LOG, Classifier (f) LOG, Classifier (g) LOG, Classifier (h) LOG, Classifier

Figure 6: Logistic regression on 2D data with 40% label noise.

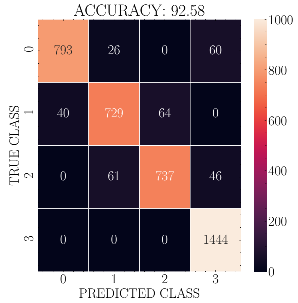
2 Neural Network for Classification on 2D Board Data

The classification task is to classify 2D features in five classes. Neural networks and support vector machines are used for classification.

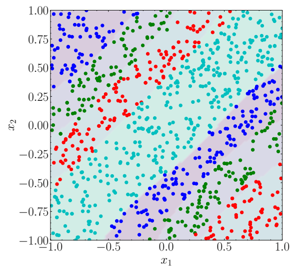
Implementation: The neural network is a feedforward network with one hidden layer with 10000 nodes. ReLU activation is used at the hidden layer and softmax activation is used at the output layer. Cross entropy is used as the loss function and training using backpropagation is done with stochastic gradient descent (SGD) and Adaptive Moment Estimation (Adam), with fixed step size and for 1000 and 100 epochs respectively. C-SVM with $C = 1$ and Gaussian kernel is used to compare against the feedforward network. The neural network and the SVM are trained with varying training sizes from 2000, 3000 and 4000, and tested on the remaining unseen samples. Training and testing is done on three datasets with varying levels of label noise.

Results: Figure 7, 8 and 9 shows classification results using neural networks and SVM on the Board dataset with 0, 10% and 25% label noise respectively. The first and third column show the accuracies in confusion matrices for training the network with 1000 epochs of SGD and 100 epochs of Adam, respectively, with varying training sizes; and the second and fourth column shows the corresponding samples with their discriminant functions. The last row shows the accuracies of the SVM with varying training sizes.

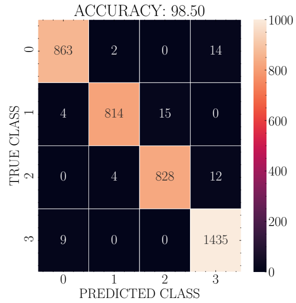
Inferences: The network trained with 1000 epochs of SGD is compared with 100 epochs of Adam. The network trained with Adam converges faster to give better accuracies. Adam is a better optimiser to use, in general. The accuracies of both the networks almost stays the same with increasing training size. 2000 samples are sufficient to represent the distributions. The accuracies of the SVM classifier are comparable to the neural networks, and in all cases better than the network trained using SGD. With the label noise increasing, the accuracies decrease, but the network trained using Adam is better than SVM, followed by the network trained using SGD.



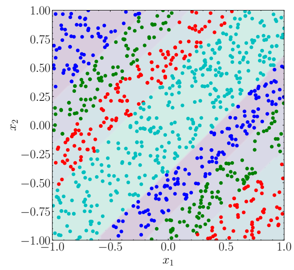
(a) SGD, Size 2000



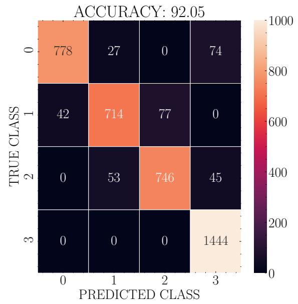
(b) SGD Classifier



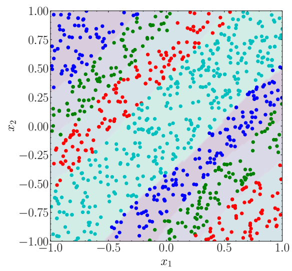
(c) Adam, Size 2000



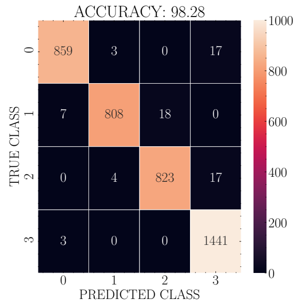
(d) Adam Classifier



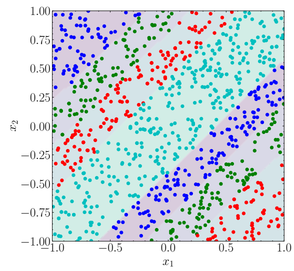
(e) SGD, Size 3000



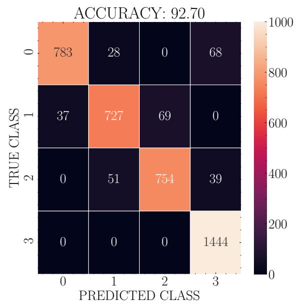
(f) SGD Classifier



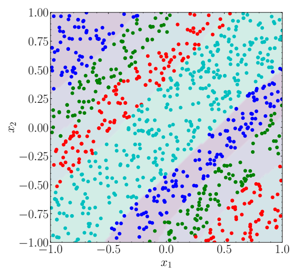
(g) Adam, Size 3000



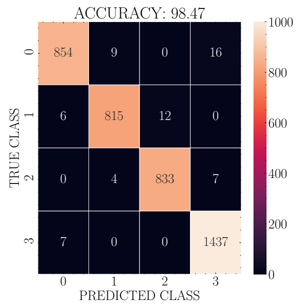
(h) Adam Classifier



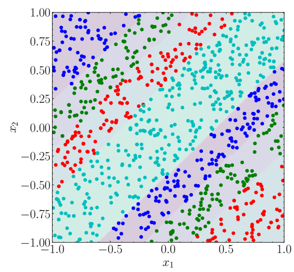
(i) SGD, Size 4000



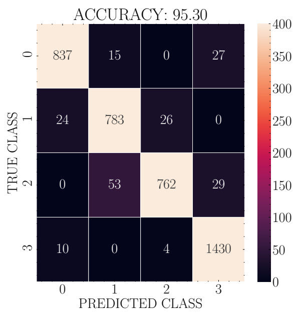
(j) SGD Classifier



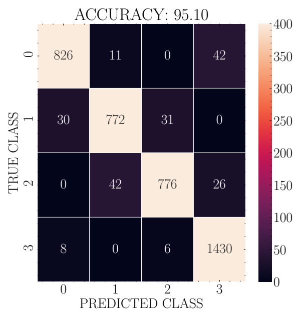
(k) Adam, Size 4000



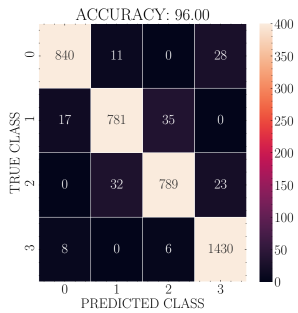
(l) Adam Classifier



(m) SVM, Size 2000

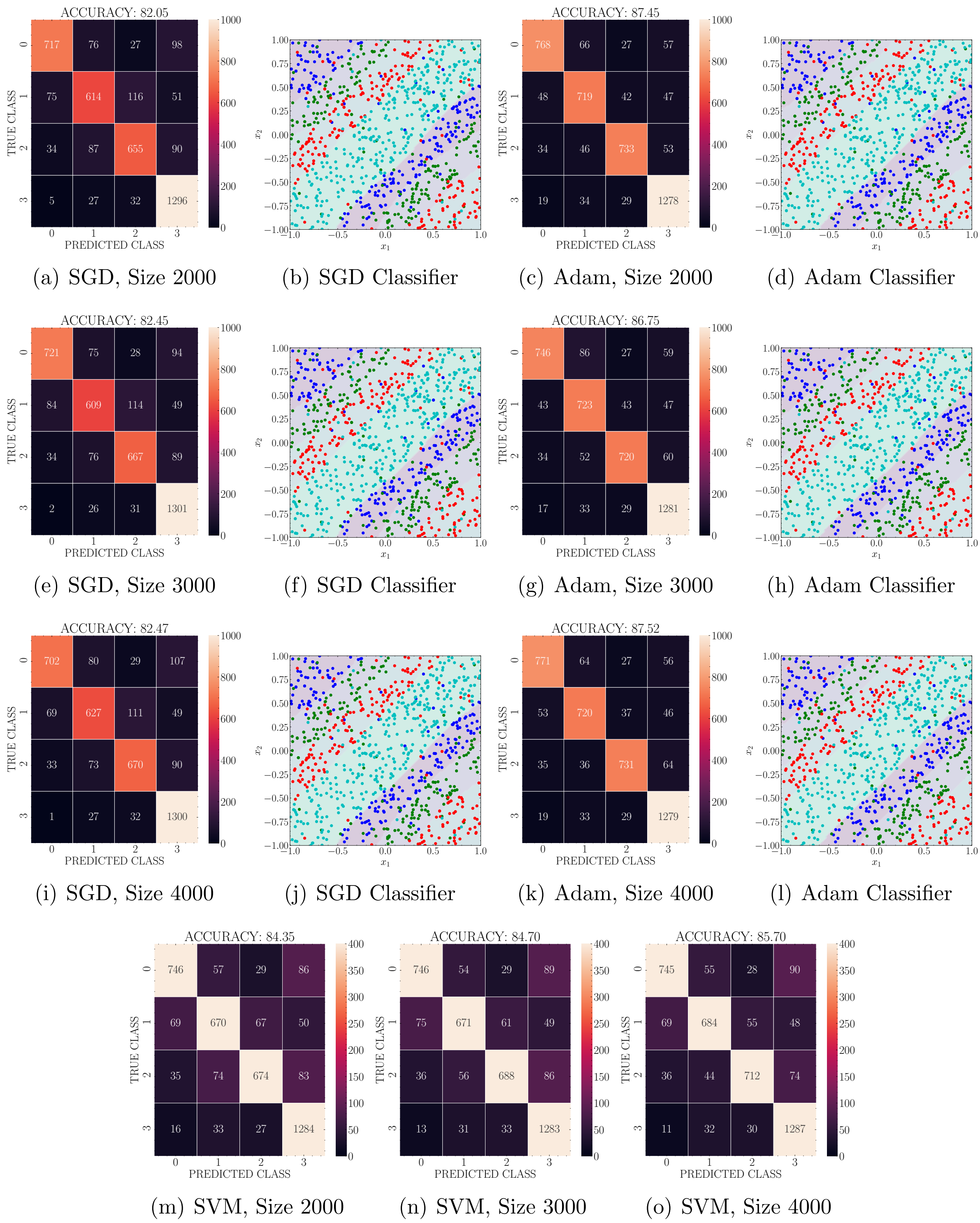


(n) SVM, Size 3000



(o) SVM, Size 4000

Figure 7: Feedforward network on 2D data with no label noise.



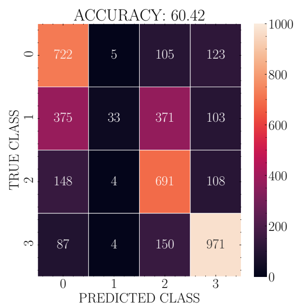
(a) SGD, Size 2000 (b) SGD Classifier (c) Adam, Size 2000 (d) Adam Classifier

(e) SGD, Size 3000 (f) SGD Classifier (g) Adam, Size 3000 (h) Adam Classifier

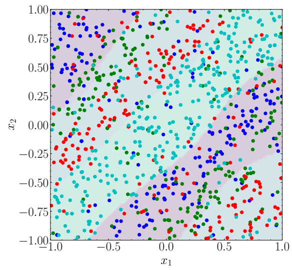
(i) SGD, Size 4000 (j) SGD Classifier (k) Adam, Size 4000 (l) Adam Classifier

(m) SVM, Size 2000 (n) SVM, Size 3000 (o) SVM, Size 4000

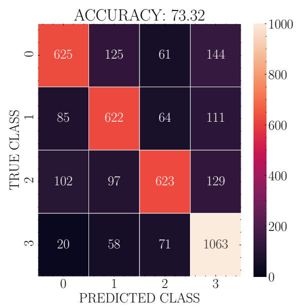
Figure 8: Feedforward network on 2D data with 10% label noise.



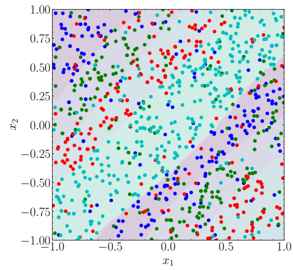
(a) SGD, Size 2000



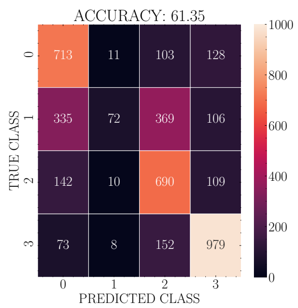
(b) SGD Classifier



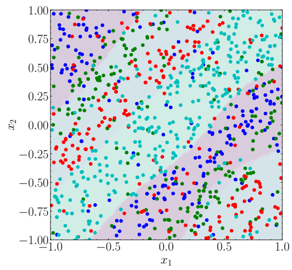
(c) Adam, Size 2000



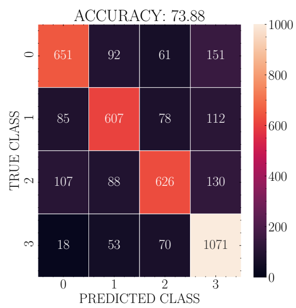
(d) Adam Classifier



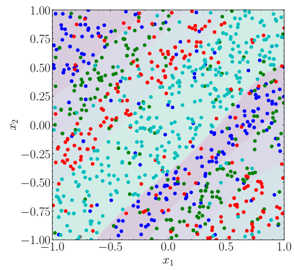
(e) SGD, Size 3000



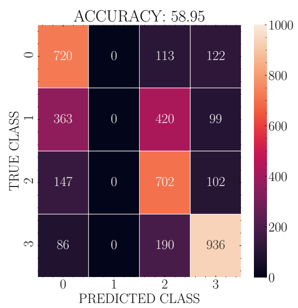
(f) SGD Classifier



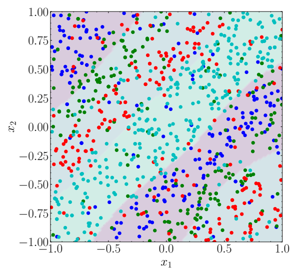
(g) Adam, Size 3000



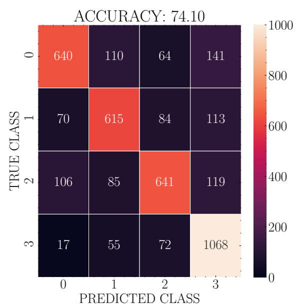
(h) Adam Classifier



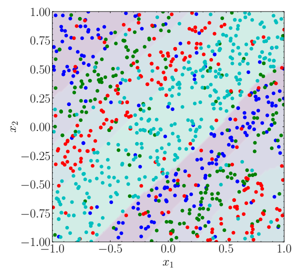
(i) SGD, Size 4000



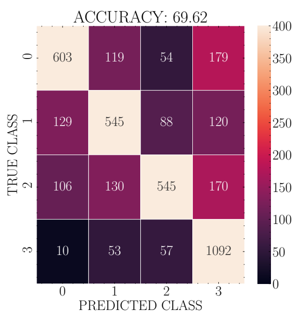
(j) SGD Classifier



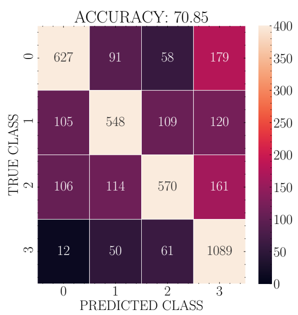
(k) Adam, Size 4000



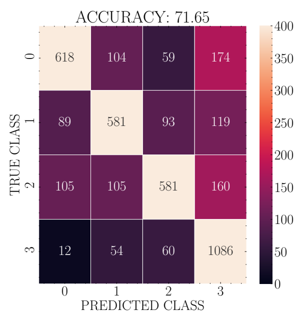
(l) Adam Classifier



(m) SVM, Size 2000



(n) SVM, Size 3000



(o) SVM, Size 4000

Figure 9: Feedforward network on 2D data with 25% label noise.

3 Convolutional Network for Classification on MNIST

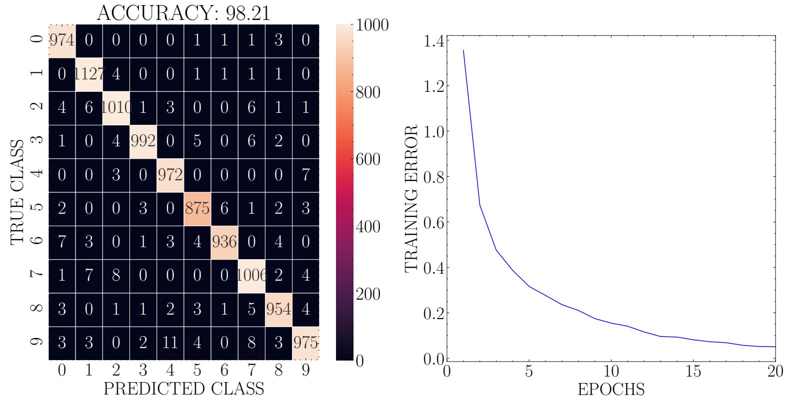
The classification task is to classify 28×28 dimensional images containing handwritten digits from $0 - 9$. A convolutional neural network and support vector machine are used for classification.

Implementation: The convolutional network used has two convolutional layers and two fully connected layers. The first convolutional layer has 6 filters of size 3×3 and the second convolutional layer has 12 filters of size 3×3. The second convolutional layer is followed by a max-pool layer of size 2×2 and stride 2.The fully connected layers have sizes 1728 and 84 respectively. All the layers have the ReLU activation function and the cross-entropy loss is used. Backpropagation is done using the Adam optimiser with constant step size and momentum weights for 20 epochs. The C-SVM with Gaussian kernel is trained with $C = 1$. The SVM is made to classify multiple classes using the 'one vs one' strategy and picking the class with most votes. The network is trained using all the training samples and tested on the complete testing data.

Results: Figures 10 and 11 shows the accuracies in confusion matrices for classification of MNIST and MNIST-ROT datasets using the same SVM and the same convolutional network. The convolutional network performs well with accuracies of 98%, and training error smoothly decreasing to zero. The SVM performs comparably to the convolutional network.

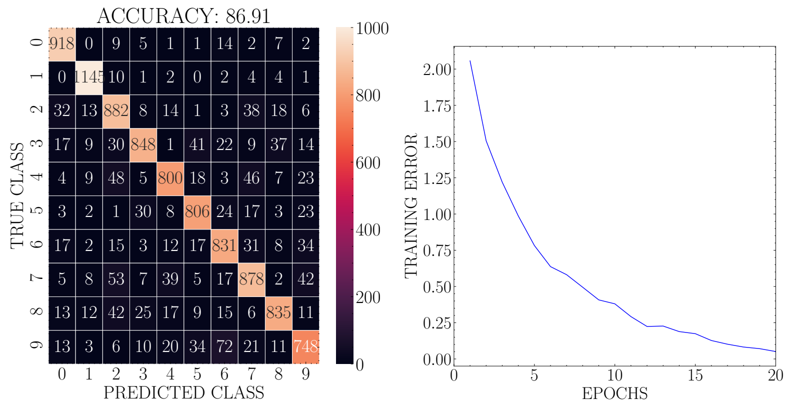
The accuracy decreases when the same network is used to classify MNIST-ROT. The MNIST-ROT dataset contains images that come from a larger class and the network that fits the MNIST distribution is not sufficient to capture the larger class of MNIST-ROT with the same amount of training. The SVM performs worse in this case.

The convolutional network on MNIST provides reasonable results, with meaningful misclassifications. For example, "zeroes" are misclassified as "eights" most often as both have closed loops. Misclassifications with SVM are more random.



(a) CNN with Adam (b) CNN Loss Function (c) SVM with RBF Kernel

Figure 10: Convolutional neural network and support vector machines for MNIST classification.



(a) CNN with Adam (b) CNN Loss Function (c) SVM with RBF Kernel

Figure 11: Convolutional neural network and support vector machines for MNIST-ROT classification.

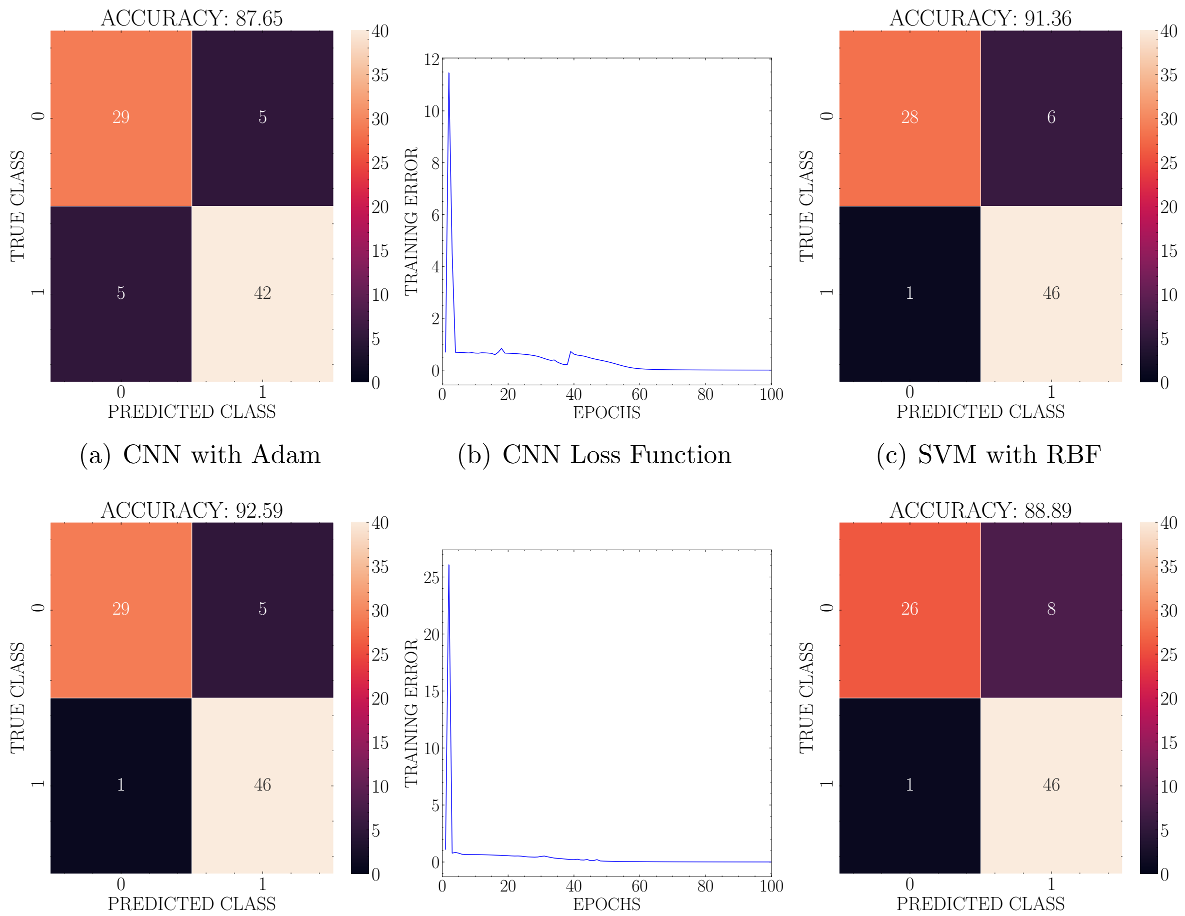
4 Convolutional Network for Classification on fMRI

The classification task is to classify fMRI recordings over time of certain parcellations to make an image, into normal and Alzheimer's disease. A convolutional network and SVM are used for classification.

Implementation: The convolutional network used has four convolutional layers and two fully connected layers. The first convolutional layer has 3 filters of size 3×3, the second convolutional layer has 6 filters of size 5×5, the third convolutional layer has 12 filters of size 5×5 and the fourth convolutional layer has 6 filters of size 3×3. The second, third and fourth convolutional layers are followed with max-pool layers of sizes 2×2, 2×2 and 4×4, respectively. The fully connected layer has 256 nodes before giving 2 outputs with softmax layer. All other layers have the ReLU activation function. The network is trained using the Adam optimiser with constant step size and momentum weights for 100 epochs. Since the number of samples in each class size are small ~ 40, the network is trained using 20% of the samples and tested on all the samples. The C-SVM with Gaussian kernel is trained with $C = 1$.

Results: Figure 12 shows the accuracies in confusion matrices for classification using the convolutional network and SVM. The first row shows the accuracies and the loss function of the CNN for the `aal` parcellation and the second row shows the accuracies and the loss function of the CNN for the `power` parcellation.

Inferences: The loss function of the CNN for both the parcellations decrease eventually, i.e., the training samples are learnt well. However, this is not directly translated to the testing class and the accuracies are only in the higher 80%. This maybe attributed to the small training and testing sizes. The SVM, in general, performs comparably to the CNN.



(a) CNN with Adam (b) CNN Loss Function (c) SVM with RBF

(d) CNN with Adam (e) CNN Loss Function (f) SVM with RBF

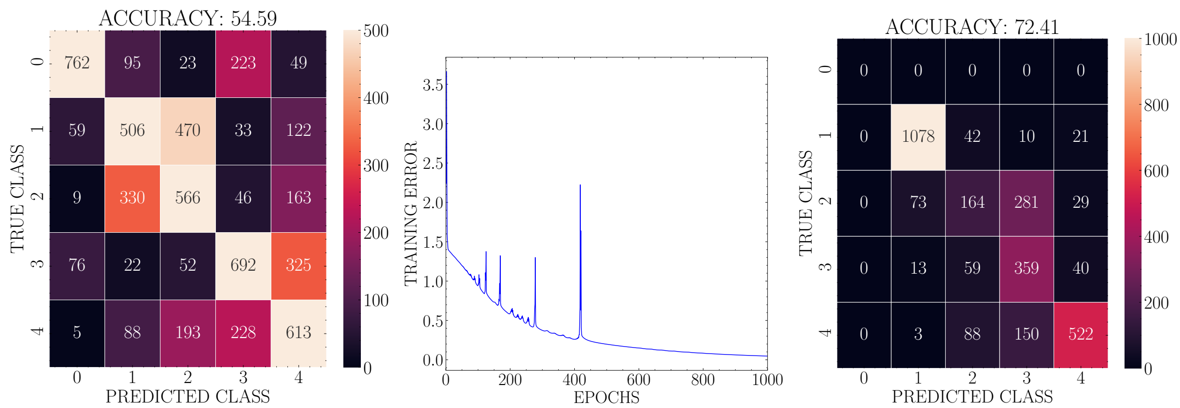
Figure 12: Convolutional neural network and support vector machines for fMRI classification.

5 Convolutional Network for Classification of EEG

The classification task is to classify EEG recordings in $\mathbb{R}^{178}$ into 5 classes. A convolutional neural network and SVM with Gaussian kernel are used for classification.

Implementation: The convolutional network used has three convolutional layers and two fully connected layers. The first convolutional layer has 3 filters of size 3, the second convolutional layer has 6 filters of size 5, and the third layer has 3 filters of size 3. The fully connected layer has 128 nodes before classifying into 5 classes. The output layer has the softmax activation and all other layers have ReLU activation. Cross entropy is used as the loss function and backpropagation is done using Adam optimiser for 1000 epochs. The SVM with RBF kernel is trained with $C = 1$. The dataset is equally divided into training and testing samples.

Results: Figure 13 shoes the accuracies in the confusion matrices for the CNN and the SVM and the loss function of the CNN.



(a) CNN with Adam (b) CNN Loss Function (c) SVM with RBF

Figure 13: Convolutional neural network and support vector machines for EEG classification.

Inferences: The CNN converges in 1000 epochs as seen by the decreasing training error, but the accuracies on the testing set are poor and compare to a coin toss.The CNN fails to capture distribution. The SVM performs better compared to the CNN, with accuracies just above 70%. The CNN outperforms the SVM in all other datasets used in the previous problems, yet does not learn the EEG samples.

6 Code Repository

The Python codes to reproduce the results can be found in the GitHub repository `https://github.com/kamath-abhijith/Nonlinear_Models`. Use `requirements.txt` to install the dependencies and the shell scripts to generate the figures.