Yameen Ajani (110096721)
Mohammed Farhan Baluch (110093799)
Jasmin Patel (110095248)

Neural Network and Deep Learning

First Project

Contents

1 Introduction

1.1 Dataset Description

The ”Dataset for Sensorless Drive Diagnosis” is a dataset from the UCI Machine Learning Repository commonly used in machine learning research for industrial applications, specifically in sensorless drive diagnosis. The dataset consists of over 58,000 records of motor sensor signals collected from a three-phase induction motor in various operating conditions.

The aim of this dataset is to develop machine learning models that can accurately diagnose faults in the motor drive system, such as broken rotor bars, bearing faults, and air gap eccentricity. The dataset includes 48 features derived from the stator current, voltage, and phase angle signals and a target variable indicating the type of fault present in the motor. The target variable has 11 possible values, each corresponding to a different fault condition.

The "Dataset for Sensorless Drive Diagnosis" is a valuable resource for researchers and practitioners working in the field of industrial automation and predictive maintenance. By accurately diagnosing faults in motor drive systems, machine learning models can help improve the reliability and safety of industrial machinery, reduce downtime and maintenance costs, and enhance overall operational efficiency.

This dataset has been used in a number of studies and research projects, and it remains a popular benchmark dataset for developing and evaluating machine learning models for sensorless drive diagnosis. Due to the complexity and variability of the signals in the dataset, it poses a significant challenge for machine learning algorithms, making it an ideal choice for testing and comparing different modelling techniques. Overall, the "Dataset for Sensorless Drive Diagnosis" provides a valuable resource for researchers and practitioners looking to develop more accurate and reliable diagnostic tools for industrial systems.

1.2 Dataset Exploration

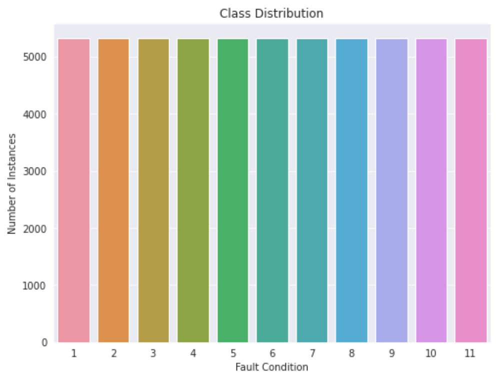


Figure 1: Class distribution bar chart

It can be observed that the classes are balanced.

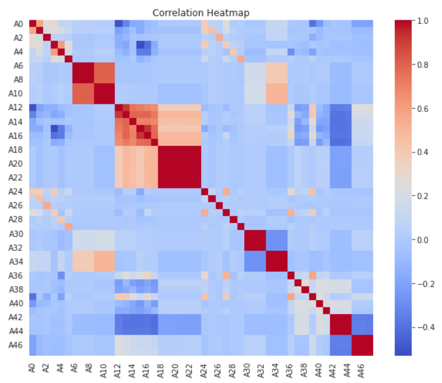


Figure 2: Correlation Matrix

The resulting heatmap shows the correlation matrix for the "Sensorless Drive Diagnosis" dataset. The darker red shades represent strong positive correlations, while the darker blue shades represent strong negative correlations. The white cells represent features that are uncorrelated.

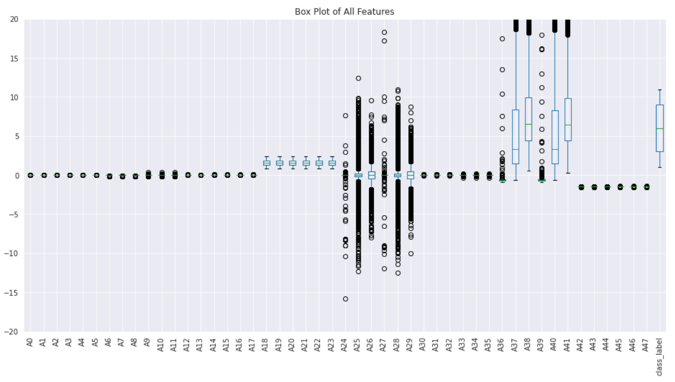


Figure 3: Boxplot of all input features

The box plot can show the central tendency, spread, and skewness of the distribution of values for each feature. The box in the plot represents the data's interquartile range (IQR), which includes the middle 50% of the values. The line in the middle of the box represents the median value of the feature.

2 Code Explanation

This code uses the Pandas library to read a dataset from a URL and store it in a DataFrame object.

The pd.read_csv function is used to read the data. The sep argument is set to " " (a space) to indicate that the data is separated by spaces. The header argument is set to None to indicate that the data has no header row.

The train_test_split function is used to split the data into training and testing sets. The X variable contains all the features of the dataset, while the y variable contains the labels. The test_size argument is set to 0.2, which means that 20% of the data will be used for testing, and 80% will be used for training.

The neural network model is built using the Keras API of TensorFlow.

2.1 Base Model (Model 1)

The base model is a simple neural network consisting of two hidden layers with 64 and 32 units, respectively, and an output layer with 11 units (corresponding to the 11 classes in the dataset). The input layer has 48 units, which is the number of features in the Sensorless Drive Diagnosis dataset. The activation function used for the hidden layers is ReLU, while the output layer uses softmax. The model is trained using categorical cross-entropy loss and stochastic gradient descent (SGD) as the optimization algorithm.

2.2 Dropout Model (Model 2)

The dropout model is a variation of the base model that uses dropout regularization to prevent overfitting. Dropout randomly sets a fraction of the input units to zero during each training iteration, which can help the model learn more robust features and reduce co-adaptation between neurons. In this case, dropout with a rate of 0.5 is applied to the base model's first and second hidden layers.

2.3 Batch Normalization Model (Model 3)

The batch normalization model is another variation of the base model that uses batch normalization to accelerate training and improve generalization. Batch normalization normalizes the activations of each hidden layer over the mini-batch, reducing the internal covariate shift problem and making the optimization landscape smoother. In this case, batch normalization is applied to the outputs of the base model's first and second hidden layers.

2.4 Regularization Model (Model 4)

The regularization model is a base model variant that uses L2 weight decay regularization to penalize large weights and prevent overfitting. L2 regularization adds a penalty term to the loss function that is proportional to the square of the L2 norm of the weight matrix. This encourages the model to learn simpler representations and can reduce the risk of overfitting. In this case, L2 regularization with a weight decay coefficient of 0.01 is applied to the weight matrices of the base model's first and second hidden layers.

2.5 Hybrid Model

The hybrid model is a combination of the dropout, batch normalization, and regularization models. Specifically, the model uses dropout with a rate of 0.5 on the first and second hidden layers, batch normalization on the outputs of the first and second hidden layers, and L2 weight decay regularization with a coefficient of 0.01 on the weight matrices of the first and second hidden layers. The activation function and loss function are the same as in the base model, and the Adam optimizer is used for training.

2.6 Testing different optimizers

The code trains the hybrid model defined earlier on different optimization algorithms (SGD, Adam, RMSprop, Adagrad, Adadelta, Adamax, Nadam) and records the loss and accuracy of the trained models on a test set. The code imports necessary libraries, including the optimization algorithms to be tested. Then, an empty dictionary called results is defined to store the loss and accuracy of the trained models on the test set. Then, The code loops through each optimization algorithm.

3 Results and discussion

3.1 Base Model (Model 1)

The base model used the Adam optimizer and no regularization techniques. It achieved a relatively high training accuracy of 0.9618 and a testing accuracy of 0.9466. The training loss was 0.1094, and the testing loss was 0.1626. The model performed well on the testing set, but the training loss is lower than the testing loss, which indicates that the model might have overfit the training set. This model did not use any regularization techniques, which may have contributed to the overfitting.

3.2 Dropout Model (Model 2)

The dropout model used the Adam optimizer and dropout regularization with a rate of 0.5. It achieved a very low training accuracy of 0.4333 and a testing accuracy of 0.2833. The training loss was 1.4061, and the testing loss was 2.1931. This model performed poorly, indicating that the high dropout rate of 0.5 may have been too severe and prevented the model from learning useful features from the data.

3.3 Batch Normalization Model (Model 3)

The batch normalization model used the Adam optimizer and batch normalization. It achieved a high training accuracy of 0.9364 and a testing accuracy of 0.9496. The training loss was 0.1718, and the testing loss was 0.2388. This model performed well, and the testing loss was slightly higher than the training loss, indicating that the model was not overfitting the data.

3.4 Regularization Model (Model 4)

The regularization model used the Adam optimizer and L2 regularization with a weight of 0.01. It achieved a training accuracy of 0.6057 and a testing accuracy of 0.6138. The training loss was 1.0977, and the testing loss was 1.0725. This model did not perform well, and both the training and testing loss were high. The use of L2 regularization may have prevented the model from learning complex features from the data.

3.5 Hybrid Model

This model used a combination of batch normalization, dropout regularization, and L2 regularization techniques. It achieved a relatively high training accuracy of 0.782 and a testing accuracy of 0.8441. The training loss was 0.6022, and the testing loss was 0.4877. The model performed better than the base and regularization models but was not as good as the batch normalization model. Combining batch normalization, dropout regularization, and L2 regularization techniques helped the model learn useful features from the data and avoid overfitting.

3.6 Summarized Comparison

In summary, the different models performed differently on the Sensorless Drive Diagnosis dataset. Table 1 shows the outline of the results, specifically accuracy and loss, of the different models tested.

Methods	Epoches	Training Loss	Training Accuracy	Testing Loss	Testing Accuracy
Model 1	50	0.1094	0.9618	0.1626	0.9466
Model 2	50	1.4061	0.4333	2.1931	0.2833
Model 3	50	0.1718	0.9364	0.2388	0.9496
Model 4	50	1.0977	0.6057	1.0725	0.6138
Hybrid	50	0.6022	0.782	0.4877	0.8441

Table 1: Comparison of accuracy and loss of different models

It is also important to note that the hybrid model achieved a test accuracy of 84.41%, which is significantly better than the regularization model, but still lower than the base model. This suggests that the combination of dropout, batch normalization, and regularization can improve the performance of the model, but the specific hyperparameters and architecture need to be carefully chosen for the particular dataset. The analysis suggests that architecture and hyperparameters play a crucial role in determining the performance of deep learning models.

3.7 Graphical comparison

In this report, we evaluated different models of neural networks using various evaluation metrics including accuracy curves, loss curves, confusion matrix, receiver operating characteristic (ROC) curve, and precision-recall (PR) curve. These metrics provide a comprehensive overview of a model's performance and can help identify areas for improvement.

Accuracy curves are used to visualize the accuracy of a model during training and validation over epochs. This metric helps to determine how well Model 1 and Model 3 are learning over epochs and can identify potential overfitting or underfitting. Figures 4a, 5a, 6a, 7a and 8a show the accuracy curves of the individual models.

Loss curves show the value of the loss function during training and validation over epochs. This metric helps to determine how well a model is minimizing its error and can help identify areas for optimization.

Loss of Model 2 is performing badly as the curve is not following the decreasing trend and is going towards convergence at epoch 50. Figures 4b, 5b, 6b, 7b and 8b show the loss curves of the individual models.

The confusion matrix provides information on the classification performance of a model by showing the number of true positives, true negatives, false positives, and false negatives. This metric can be used to calculate additional metrics such as precision, recall, and F1 score. We can see some light blue colored spots across the grid of the confusion matrix of Model 3, which indicates that some classes are being misclassified. Figures 4e, 5e, 6e, 7e and 8e show the confusion matrix of the respective models.

The ROC curve is used to evaluate the performance of the classifier by plotting the true positive rate against the false positive rate. This metric is useful when the cost of false positives and negatives is different. Overall, classes have a good ROC curve in the hybrid model, but class 5 and class 6 have some impurities in the classification task. But as we can see, Model 1 and Model 3 have the best ROC curves. Figures 4c, 5c, 6c, 7c and 8c show the ROC curves of the individual models.

PR curve is used to evaluate the performance of the classifier by plotting the precision against the recall. This metric is useful when the cost of false positives and false negatives is not the same and when the dataset is imbalanced. Again, through the PR curve, we can prove that Model 1 and Model 3 surpass the other models and are best for our classification task. Figures 4d, 5d, 6d, 7d and 8d show the Precision-Recall curves of the individual models.

3.8 Optimizer comparision

In this study, we evaluated the performance of different optimizers in neural network models for a classification task. We tested the optimizers SGD, Adam, RMSprop, Adagrad, Adadelta, Adamax, and NAdam. The models were trained for 50 epochs, and their training and validation accuracy and loss values were recorded. Figure 9 contains subplots showing the comparison of training accuracy, training loss, validation accuracy and validation loss for the optimizer functions tested. Our analysis revealed that SGD performed the worst among the tested optimizers, with a low training accuracy of around 35 after 50 epochs. Adam showed a slightly better performance than SGD, with a training accuracy starting at 45 and increasing to 55 after 50 epochs. On the other hand, RMSprop, Adagrad, Adadelta, Adamax, and Nadam showed a consistently high training accuracy of around 55 from the beginning of the epochs to 50 epochs. The trend in training loss followed a similar pattern, with SGD having the highest loss values and RMSprop, Adagrad, Adadelta, Adamax, and NAdam having comparatively lower loss values. Furthermore, all optimizers' validation accuracy and loss curves showed high fluctuations and a similar pattern.

4 Conclusion

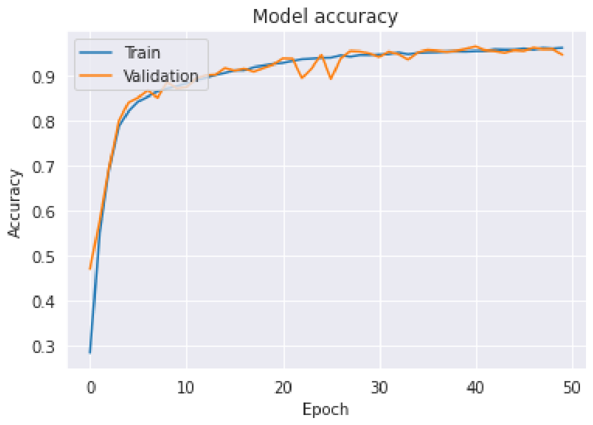
The results of our evaluation reveal that Model 1 and Model 3 have the best performance among the five models tested. These models achieved high training and testing accuracy with low loss values. On the other hand, Model 2 had the poorest performance, with low accuracy and high loss values, suggesting that the model was underfitting or built with weak architecture or incorrect hyperparameters. Model 4 and the Hybrid model had moderate performances. Our analysis highlights that architecture and hyperparameters play a critical role in the performance of deep learning models. Overall, the evaluation metrics provided a comprehensive overview of the models' performance and highlighted improvement areas. Considering these metrics when designing and evaluating deep learning models to optimize their performance is essential. Also, our findings suggest that the choice of optimizer can significantly impact the performance of neural network models and that optimizers such as RMSprop, Adagrad, Adadelta, Adamax, and Nadam can be more effective in achieving high accuracy and low loss values for a given classification task.

Work Distribution

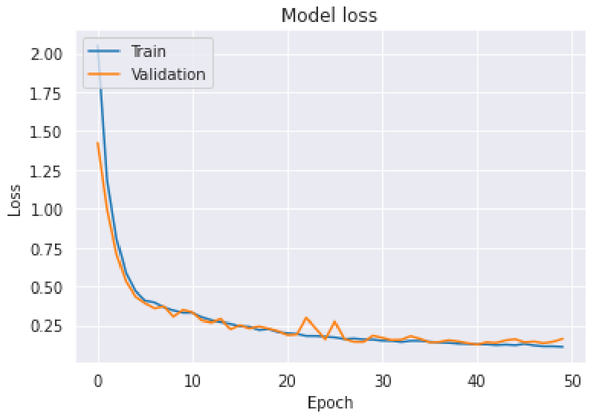
All the participants contributed equally to this project in all the sections including coding and research on relevant topics.

5 Appendix

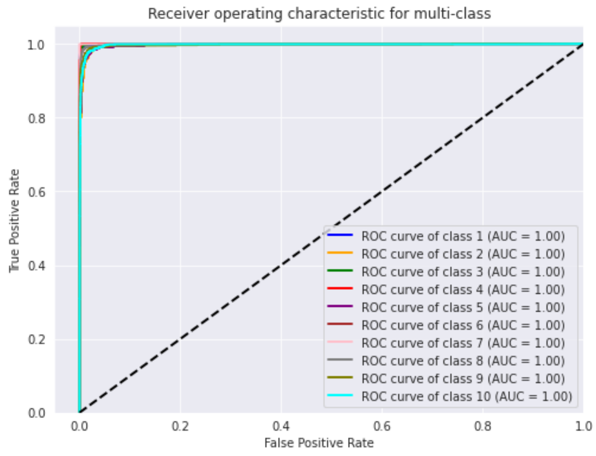
Plots - Base Model



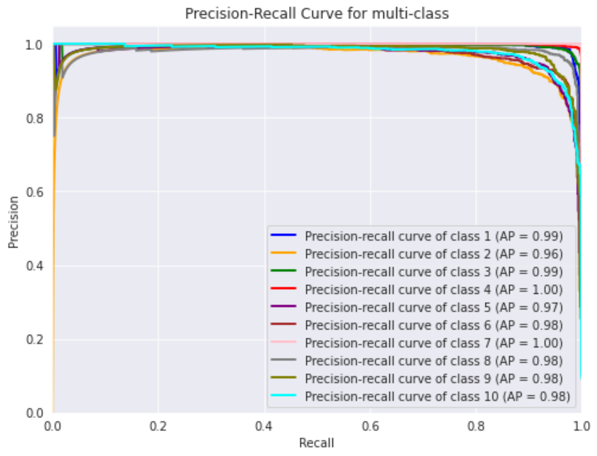
(a) Plot showing training & test accuracy of Model 1



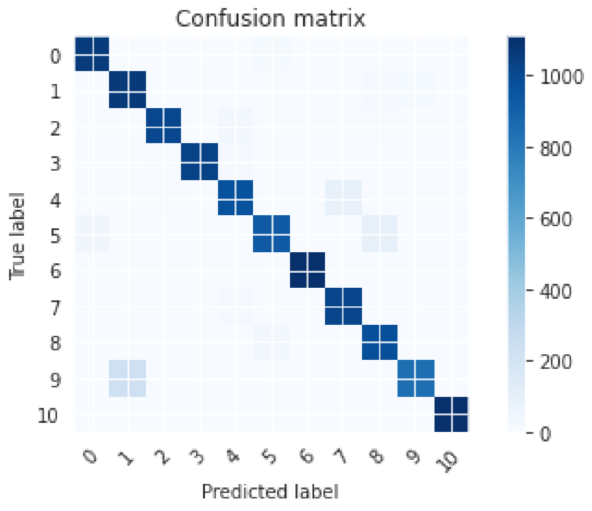
(b) Plot showing training & test loss of Model 1



(c) Plot showing ROC curve of Model 1



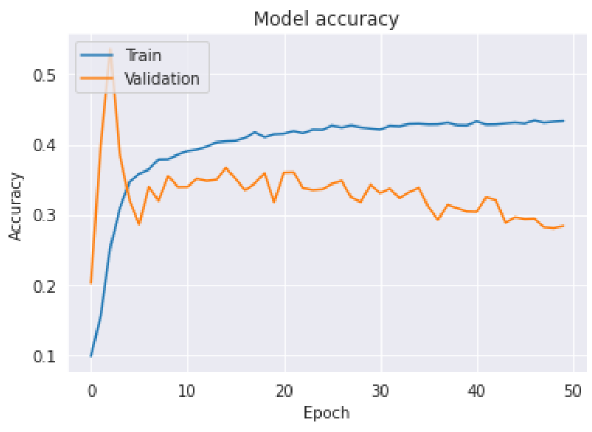
(d) Plot showing Precision-Recall Curve of Model 1



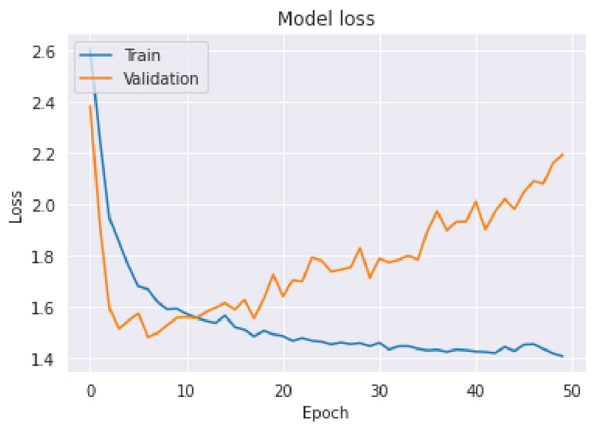
(e) Plot showing Confusion Matrix of Model 1

Figure 4: Plots for Model 1 - Base Model

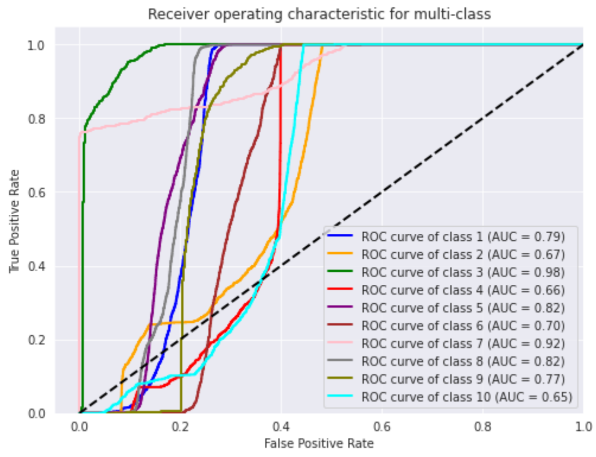
Plots - Dropout Model



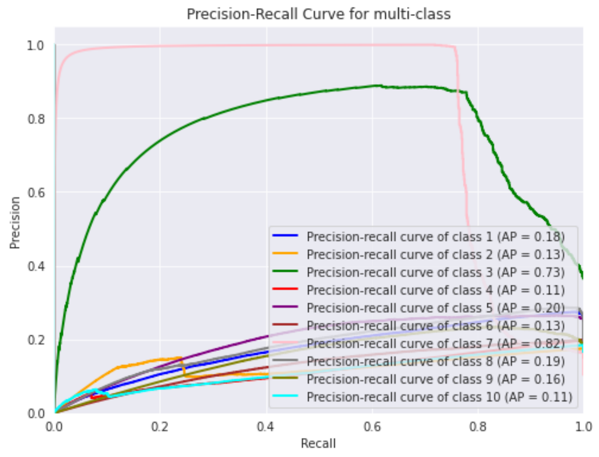
(a) Plot showing training & test accuracy of Model 2



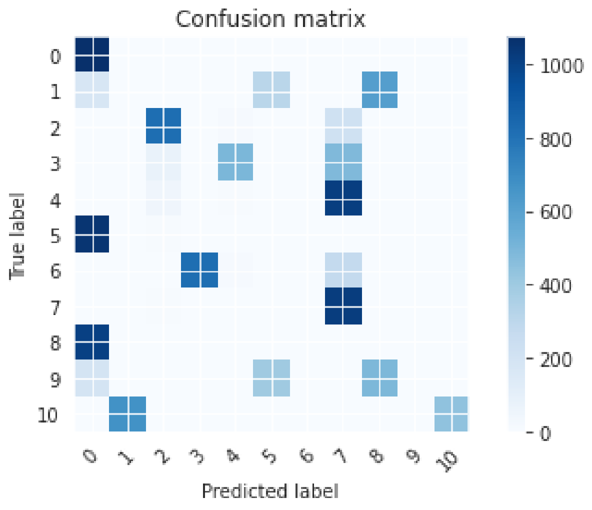
(b) Plot showing training & test loss of Model 2



(c) Plot showing ROC curve of Model 2



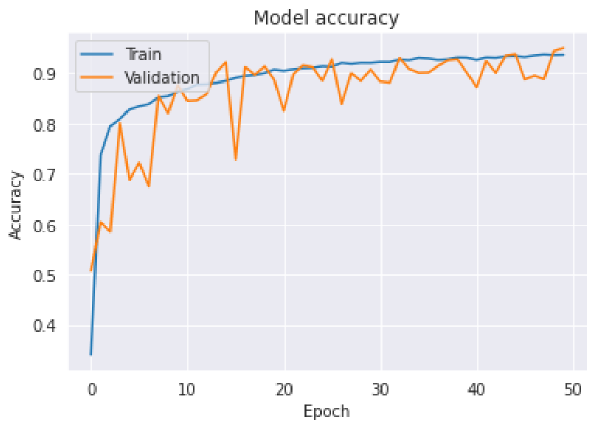
(d) Plot showing Precision-Recall Curve of Model 2



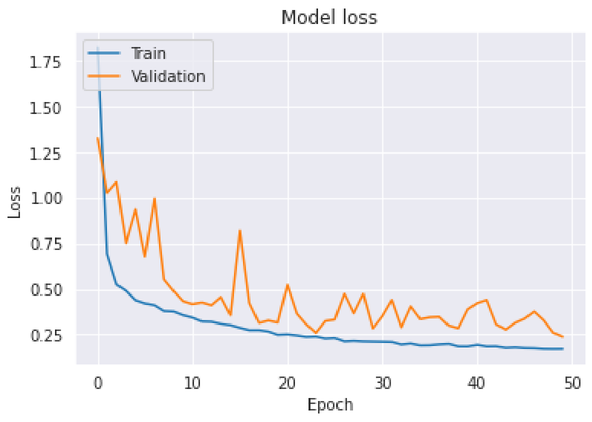
(e) Plot showing Confusion Matrix of Model 2

Figure 5: Plots for Model 2 - Dropout Model

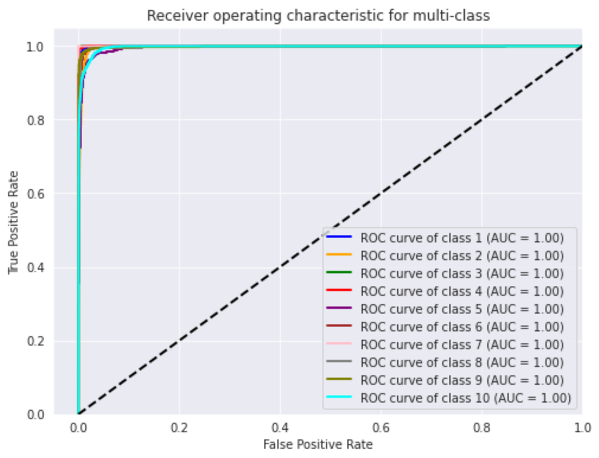
Plots - Batch Normalization Model



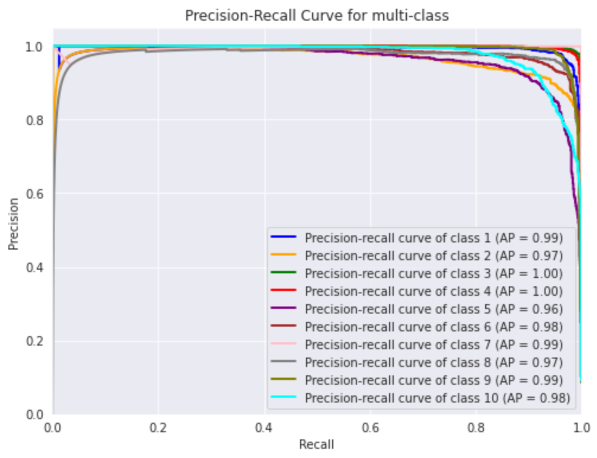
(a) Plot showing training & test accuracy of Model 3



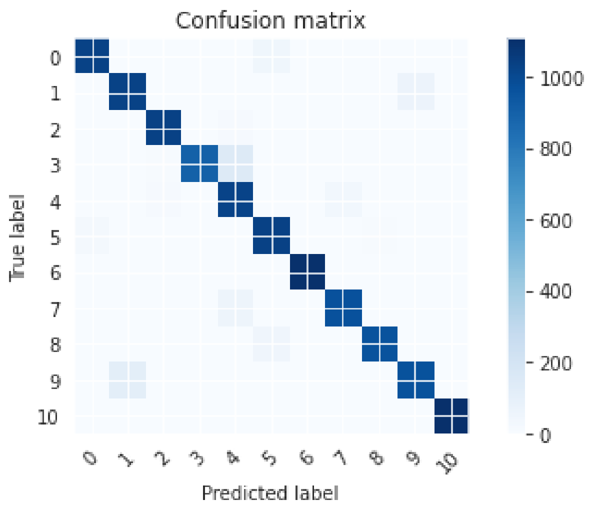
(b) Plot showing training & test loss of Model 3



(c) Plot showing ROC curve of Model 3



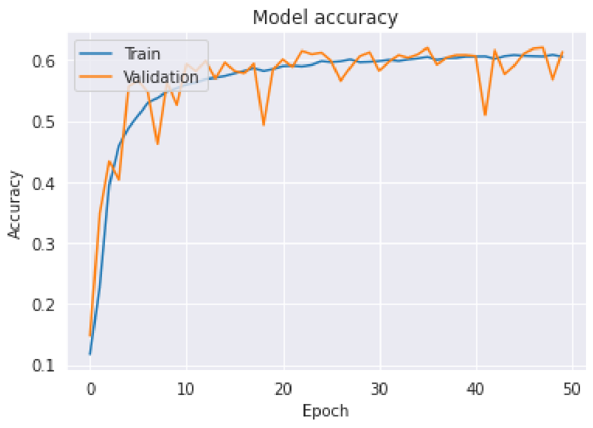
(d) Plot showing Precision-Recall Curve of Model 3



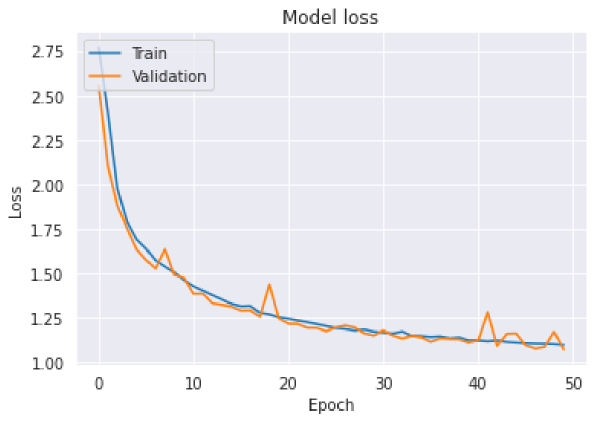
(e) Plot showing Confusion Matrix of Model 3

Figure 6: Plots for Model 3 - Batch Normalization Model

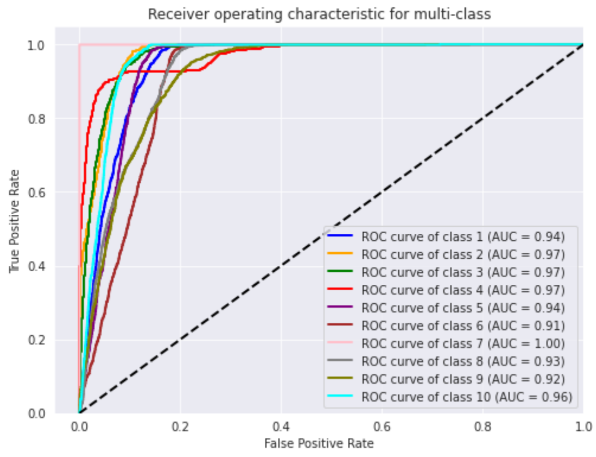
Plots - Regularization Model



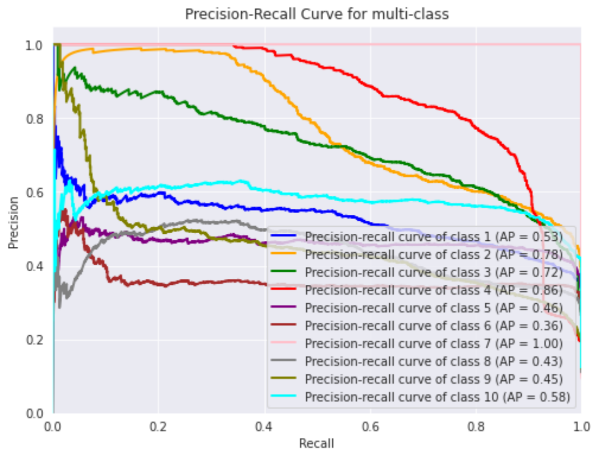
(a) Plot showing training & test accuracy of Model 4



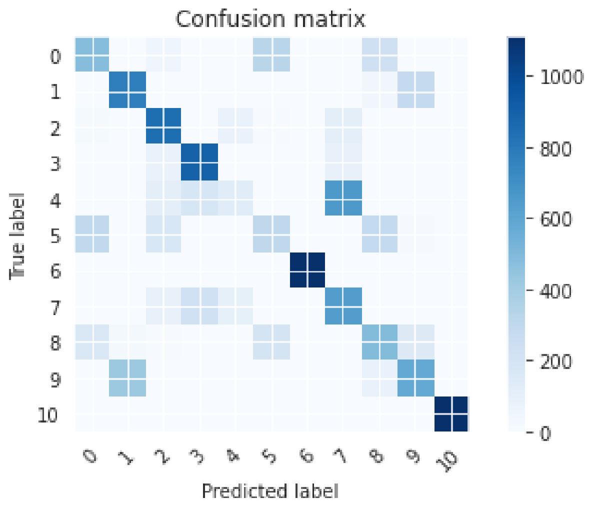
(b) Plot showing training & test loss of Model 4



(c) Plot showing ROC curve of Model 4



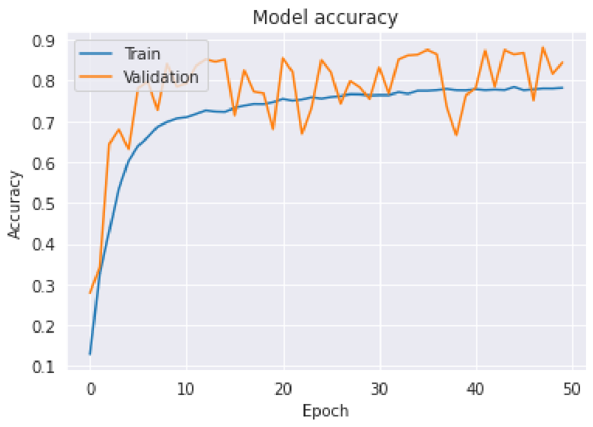
(d) Plot showing Precision-Recall Curve of Model 4



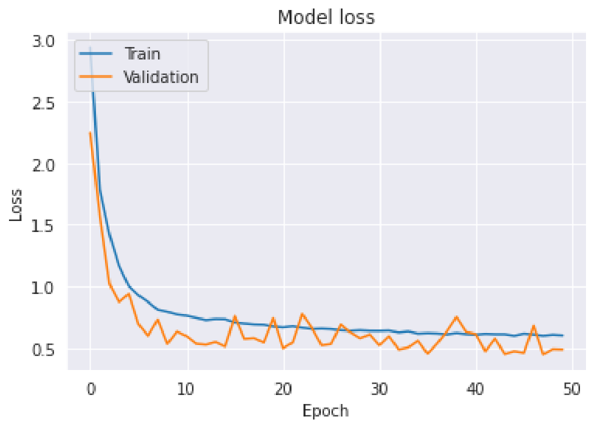
(e) Plot showing Confusion Matrix of Model 4

Figure 7: Plots for Model 4 - Regularization Model

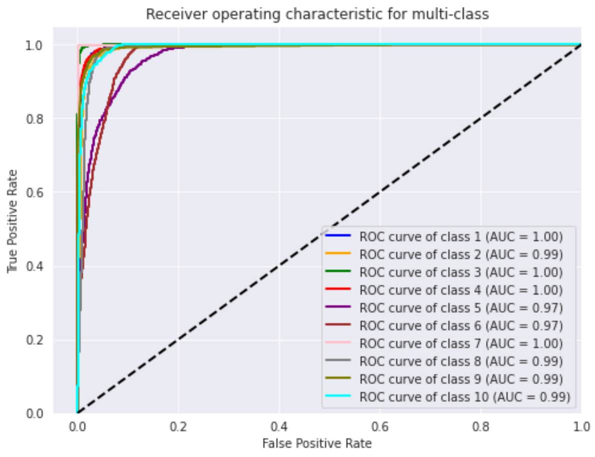
Plots - Hybrid Model



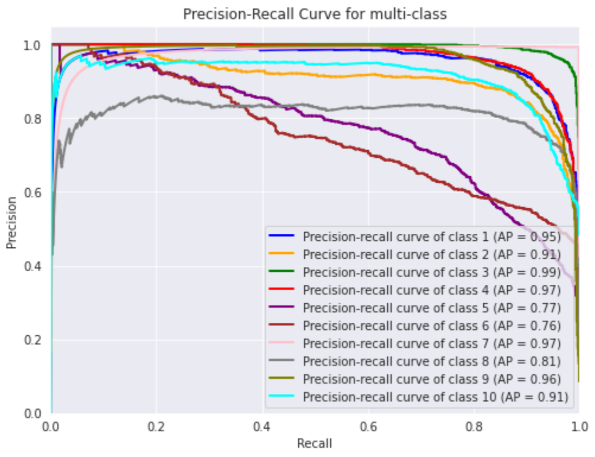
(a) Plot showing training & test accuracy of Hybrid Model



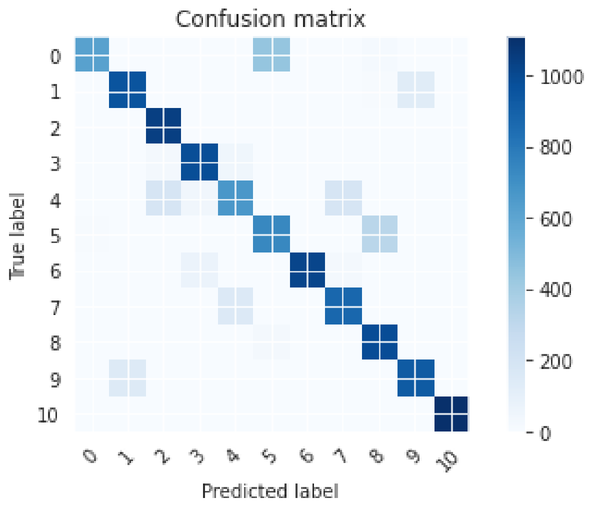
(b) Plot showing training & test loss of Hybrid Model



(c) Plot showing ROC curve of Hybrid Model



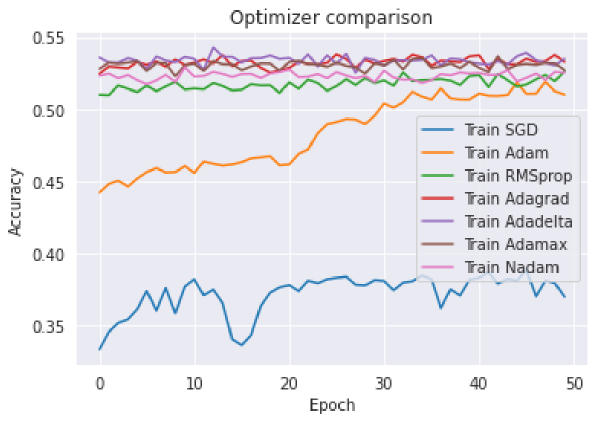
(d) Plot showing Precision-Recall Curve of Hybrid Model



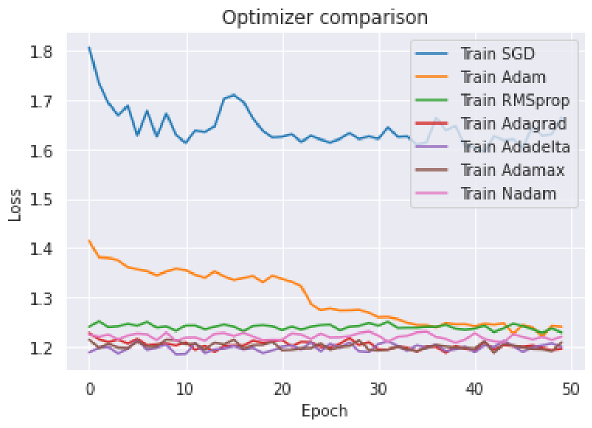
(e) Plot showing Confusion Matrix of Hybrid Model

Figure 8: Plots for Hybrid Model

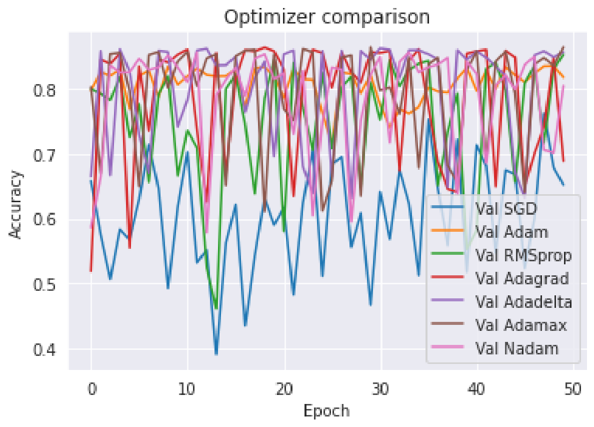
Plots - Optimizer Comparison



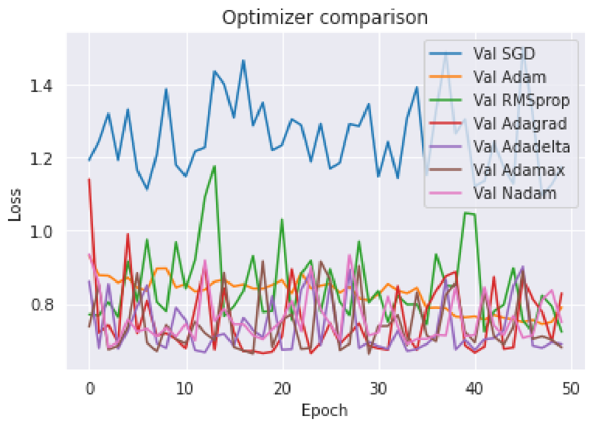
(a) Plot showing training accuracy of different optimizers



(b) Plot showing training loss of different optimizers



(c) Plot showing validation accuracy of different optimizers



(d) Plot showing validation loss of different optimizers

Figure 9: Plots for Optimizer Comparison on Base Model