Assignment 1A –

Regression, Classification & Deep Networks

CAB420 – Machine Learning

Gregory Mandall - n10757163

Submitted: April 2024

* Code adapted from CAB420 lectures and practical solutions

REGRESSION

1. Pre-processing

The ‘Communities and Crime’ dataset encompasses diverse features with varying units such as percentages, dollars, and counts. Although the dataset is normalised, it exhibits noticeable outliers in several features. This is denoted by the wide dispersion of data points from the mean, depicted in Figure 1. Additionally, a large quantity of zero values is present, potentially impacting the reliability of the dataset. Standardisation was performed by penalising coefficients based on their relative magnitudes. This approach aimed to reduce the impact of outliers and ensure that all variables contribute to the model’s predictive power in a balanced manner, regardless of their original scales, units.

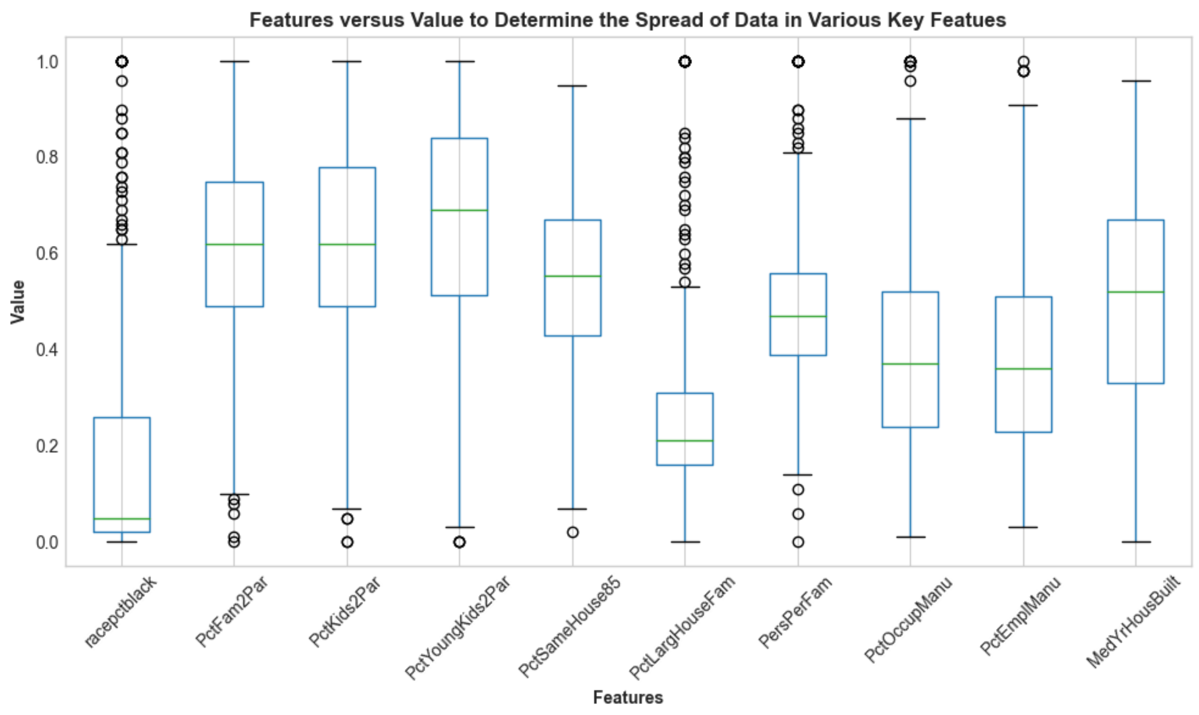


Figure 1: Various Features Contained Within the ‘Communities and Crime’ Dataset Versus Value to Determine Variable Dispersion in Various Model Parameters.

2. Model Details (Hyperparameters)

The validation dataset was used to select the optimal regularisation parameter (λ) for the Ridge and Least Absolute Shrinkage and Selection Operator (LASSO) regression models. The hyperparameter selection process involved an initial grid search using values ranging from $10^{-5} – 10^{2}$ on a logarithmic scale. The resulting lambda values were systematically evaluated, and the corresponding Root Mean Square Error (RMSE) values for the training and validation sets were recorded. Initial RMSE versus lambda values for Ridge and LASSO regression are shown in Figure 2 and Figure 3 below. Visualising RMSE versus the log of lambda assisted in identifying the lambda value corresponding with the lowest validation RMSE, indicating optimal model performance. A refined grid search was then conducted around the λ value corresponding to the validation RMSE minima depicted in Figure 2 and Figure 3 to increase hyperparameter precision. The final lambda values selected through this process are:

$$\lambda_{Ridge} = 0.063$$

$$\lambda_{LASSO} = 0.002$$

The final hyperparameter values attempt to minimise validation error and enhance the generalisation performance of the Ridge and Lasso models on unseen data.

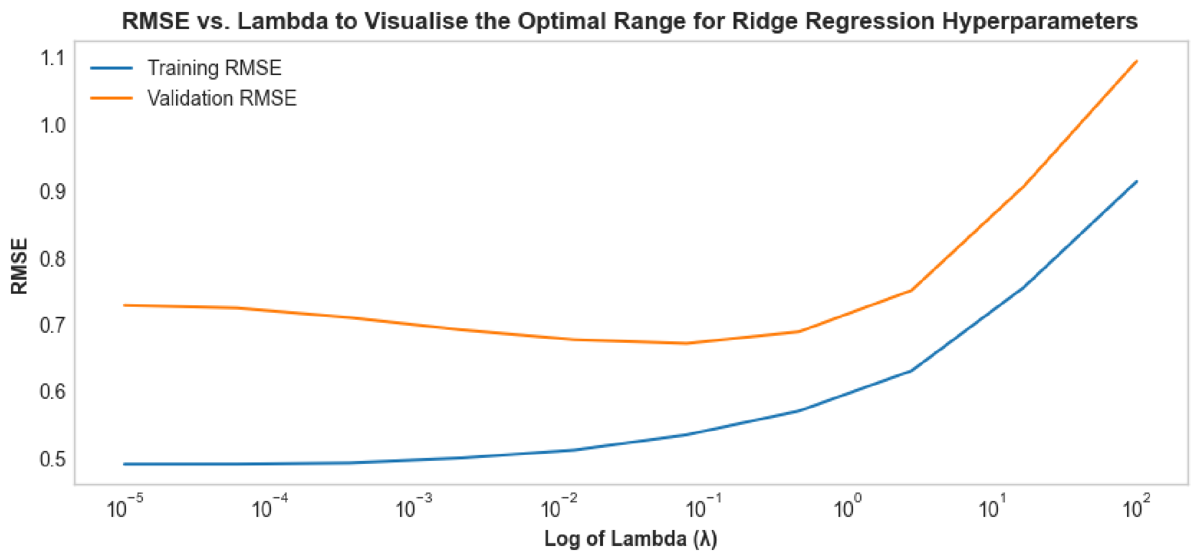


Figure 2: RMSE versus Log of Lambda to Assist in Identifying the Lambda Value Corresponding with The Lowest Validation RMSE for the Ridge Regression Model.

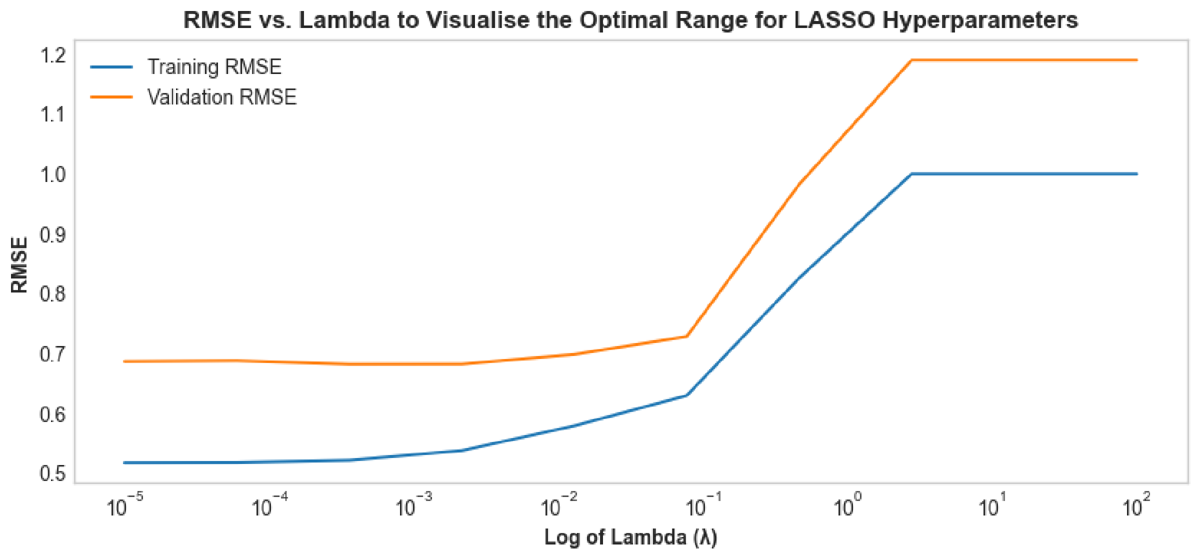


Figure 3: RMSE versus Log of Lambda to Assist in Identifying the Lambda Value Corresponding with The Lowest Validation RMSE for the LASSO Regression Model.

3. Evaluation and Analysis

The linear, ridge, and LASSO models showcase strong accuracy metrics, with the Ridge and LASSO models exhibiting regularisation benefits by offering improved generalisation. This is depicted by the slightly lower test RMSE and R^2 values in comparison to the linear model. All training and test RMSE values are shown in Table 1 below. The Ridge Regression model demonstrates a balanced performance with a training RMSE of 0.112, test RMSE of 0.131, and an R^2 value of 0.718. The LASSO model potentially further reduces overfitting, suggested by the higher training RMSE of 0.114, test RMSE of 0.134, and lower R-squared value of 0.711 relative to the ridge regression and simple linear models.

Table 1: Training RMSE, Test RMSE, and R-squared values of the Linear, Ridge, and LASSO Regression Models.

Model	**Training RMSE**	**Test RMSE**	**R^2**
Linear Model	0.104	0.153	0.759
Ridge Regression	0.112	0.131	0.718
LASSO	0.114	0.134	0.711

Inappreciable nuances between the two regularisation methods may be attributed to the distinct method ridge and LASSO regularisation used to model coefficients. Lasso regularisation facilitates increasingly compact models by enforcing coefficients to zero. Ridge regularisation penalises the magnitude of coefficients without driving them to zero. The difference in lambda values and the resulting accuracy metrics between LASSO and ridge regularisation may be attributed to their different penalty terms. The LASSO penalty term is proportional to the absolute value of the coefficients, while the ridge penalty term is proportional to the square of the coefficients. As a result, LASSO drives less important features' coefficients to zero, potentially affecting accuracy metrics such as RMSE and R2 by promoting sparsity and reducing the impact of less significant features. The improved performance on unseen data suggests that ridge and LASSO regression increases model validity relative to a simple linear model.

Residual plots were employed to evaluate the model's 'goodness of fit' and identify potential issues surrounding heteroscedasticity and abnormal residual trends. The plots shown in Figure 4, Figure 5, and Figure 6 were used as diagnostic tools to assess model validity by visualising residuals. Heteroscedasticity is observed across all models. An additional linear pattern is evident in the residuals, and several potential outliers within the distribution are also present. These outliers could potentially skew the results of the regression models, leading to reduced predictive accuracy. The linear trend in residuals could be attributed to numerous zero values present within the dataset. Although the residual plots generally indicate heteroscedasticity, systematic bias suggests limitations are present in the modelling process, potentially impacting the models' validity.

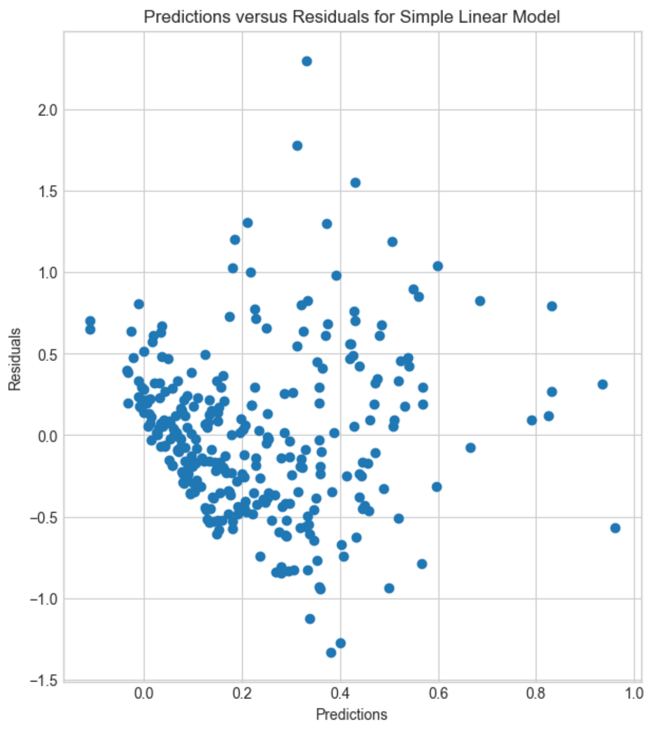


Figure 4: Predictions versus Residuals for the Simple Linear Model to Assess and Evaluate Model Validity

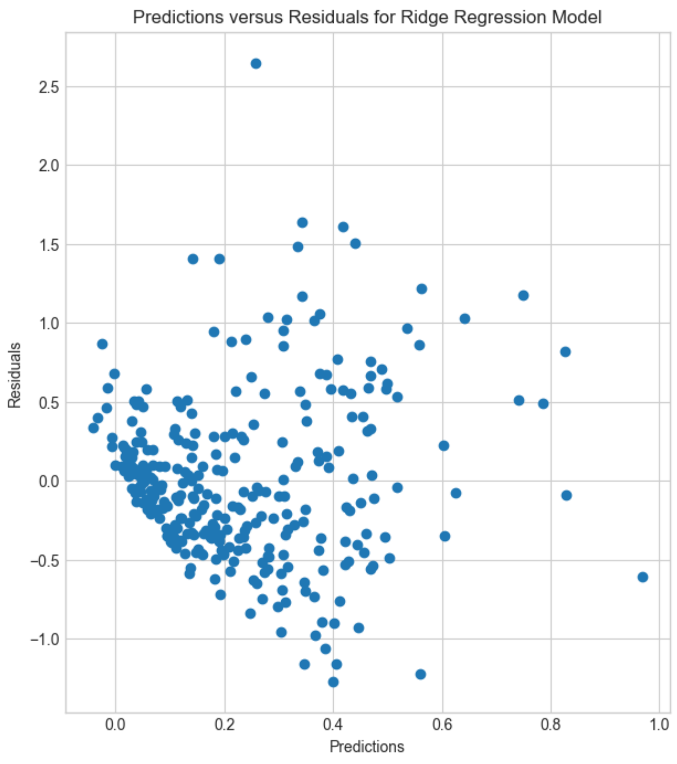


Figure 5: Predictions versus Residuals for the Ridge Regression Model to Assess and Evaluate Model Validity

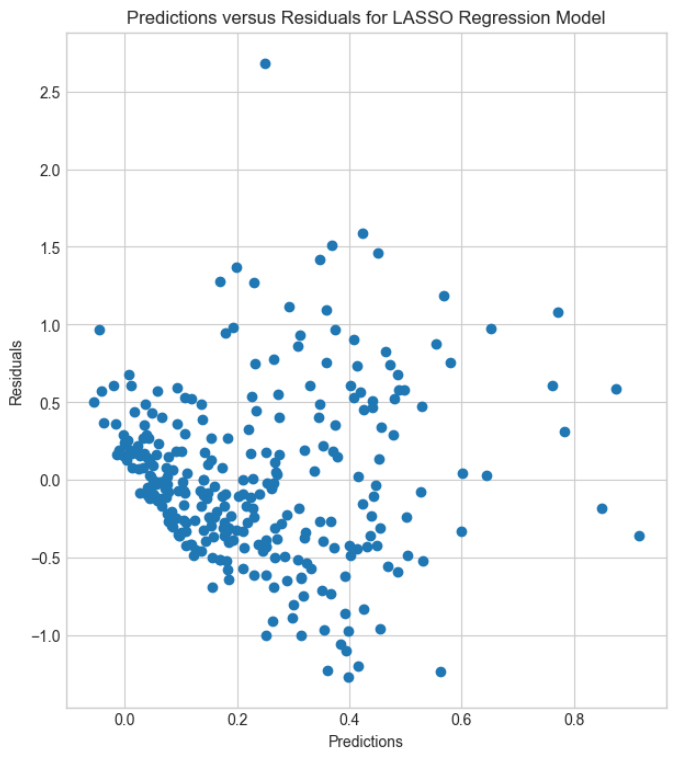


Figure 6: Predictions versus Residuals for the LASSO Model to Assess and Evaluate Model Validity.

Additional Q-Q plots were examined to assess whether the models adhere to the assumption that the residuals follow a Gaussian distribution. While the plots generally indicate conformity to a Gaussian distribution, there are noticeable deviations in the form of skewed tails in the simple linear, ridge, and LASSO models. This suggests that potential violations of the regression assumption may occur. It is also crucial to consider additional accuracy metrics and the statistical significance of the variables included. The high p-values for many variables may indicate a lack of robustness, multicollinearity issues, or relevance in the chosen features. This may affect the model's validity. Additionally, the model's accuracy may not be sufficient for making crucial socio-economic predictions. Future funding or policy decisions may require a deeper understanding of domain-specific nuances and a comprehensive approach to socio-economic data analysis using increasingly accurate, valid, and reliable machine learning models.

4. Additional Ethical Considerations

The sensitivity of socio-economic data requires assiduous evaluation to avoid any misinterpretation when predicting crime rates. A predictive model that relies on historical crime data without considering evolving socio-economic factors over time may skew the predictive accuracy of the trained model and impact its validity in real-world applications. Additional limitations, such as assumptions about data distributions or feature selection biases, need to be communicated and addressed while also considering the impact the model's predictions may have on a broader socio-economic scale. An ethical machine-learning approach in this domain involves assessing biases, limitations, and responsible data usage to achieve transparency, accountability, and contestability during model deployment. This may mitigate potential negative social impacts caused by inaccurate predictions, such as assigning a negative reputation to an area due to high crime predictions.

CLASSIFICATION

1. Pre-processing

The data set provided consists of 4 classes: d = 54 instances, h = 48 instances, o = 37 instances, and s = 59 instances. The distribution indicates potential class imbalance issues, possibly introducing challenges during model training. Standardisation was applied due to the involvement of distance measures in the CKNN and Support Vector Machine (SVM) model processes, as SVM's utilise distance-based calculations to determine a hyperplane that separates the classes in the feature space and CKNN's employ distance calculations to identify the closest neighbours for classification. Standardisation ensures that the distances computed between data points are consistent across all dimensions and ensures that all features contribute proportionally to the model's decision-making process in distance-based algorithms like SVM and CKNN. Although standardisation may directly impact models like SVM and CKNN due to their reliance on distance metrics, it doesn't directly affect Random Forests. However, standardising the data when comparing SVM, CKNN, and Random Forest models is crucial to develop an unbiased comparison across all three models.

2. Model Details (Hyperparameters)

A grid search method was utilised to select optimal hyperparameters for the CKNN, SVM, and random forest models. This consisted of a systematic approach to explore a range of specified values for each hyperparameter and evaluate each model's performance in terms of F1-score during the validation phase. The K-Nearest Neighbours model evaluated values of k from k=8 to k=198 and included disparate distance metrics such as Euclidean, Cityblock, and Cosine to reflect any potential diverse relationships present in the data. The regularisation parameter 'C' for the SVM models was varied across a logarithmic scale from 1e−5 to 1e5. The kernel types explored included Linear, Polynomial, and Radial Basis Function (RBF). Respective parameters such as gamma were set to 0.1, 1, 10, 'auto', and 'scale', and the degree was tested at 3, 5, 7, and 9. The performance of the '1vs1' and '1vsAll' methods was also examined during this phase. The Random Forest model hyperparameters included tree depth, which was tested at 4, 8, 16, and 32 and the number of estimators which varied from 100, 200, 400, and 800. Each grid search was crucial in mitigating the risk of overfitting or underfitting, given the slight class imbalance observed in the dataset. The final values of hyperparameters for each model are outlined below:

CKNN: *metric* = 'euclidean', *k* = 16

SVM: *kernel* = Linear, *c* = 0.1, (1vs1classifier)

Random Forest: *max_depth* = 8, *n_estimators* = 200

Using a value of k=16 in the CKNN model aligns with the dataset's scale and ensures sufficient local neighbourhood detail is captured while also attempting to mitigate overfitting. A smaller value of k may make the model hypersensitive to noise. A larger k value may diminish local significance by encompassing a wide-ranging neighbourhood, potentially causing the model to overlook underlying patterns in the data. Employing a smaller c value in the SVM enhanced model generalisation. A larger c value led to increased overfitting. This may be caused by the model's attempts to learn increasingly accurate predictions for the training data. Setting the max depth of the random forest larger than 8 may capture an increased amount of data specifics. However, this risks overfitting by fitting to noise rather than underlying data patterns. A smaller value for the max depth may fail to capture an adequate amount of data specifics. A max depth of 8 and setting the number of estimators at 200 ensures enough complexity to model interactions effectively, considering the size of the dataset.

3. Evaluation and Analysis

A weighted average F1 score was used to assess each model's performance on the test set, as class frequency variations may potentially skew the accuracy metric. The CKNN model achieved an F1 score of 0.785 and showed varying effectiveness across classes. The model seems to struggle to predict the least represented class 'o'. This suggests that CKNN may face difficulties when fewer instances are available to define the class boundary, potentially due to its reliance on local data structures. CKNN's sensitivity to outliers or potential class imbalances may also be attributed to its moderate performance relative to the SVM and random forest models. The confusion matrix representing the CKNN model's performance and F1 score is shown in Figure 7 below.

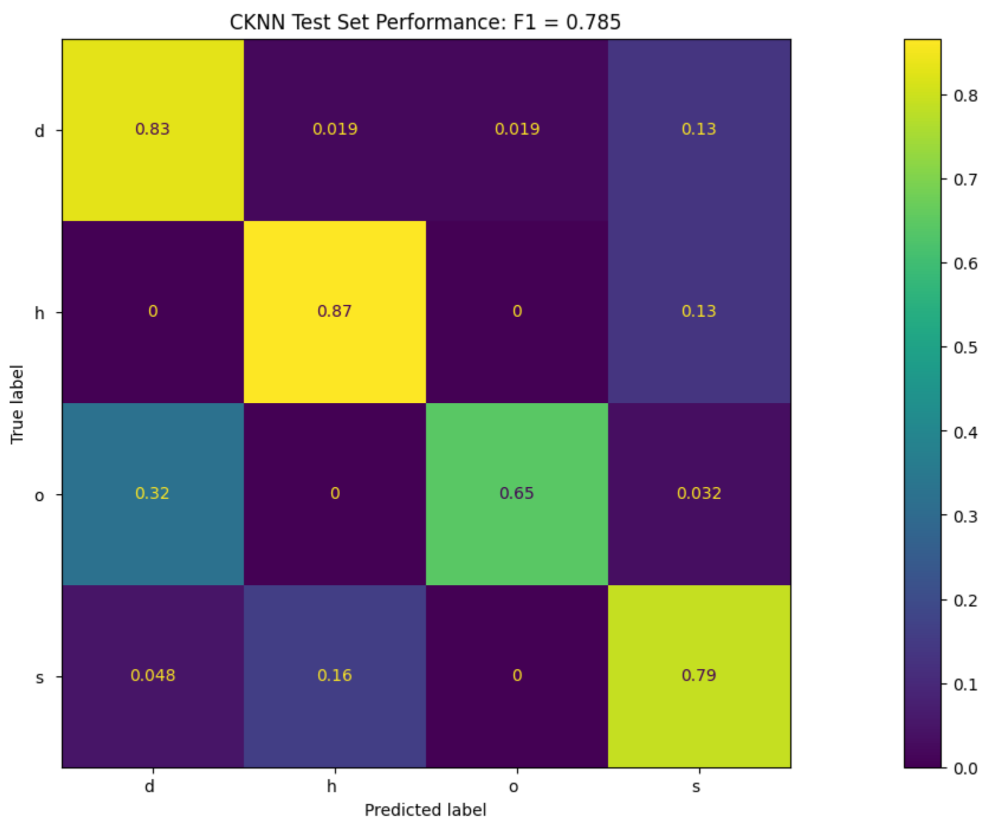


Figure 7: Confusion Matrix to Visualise the Performance of the CKNN model on the Test Data Set, and to Compare its Performance to the SVM and Random Forest Models.

The SVM model attained the highest F1 score of 0.843 and demonstrated a robust performance across all classes relative to the CKNN and random forest models. The high recall across classes suggests that the linear kernel combined with appropriate hyperparameters managed to achieve an adequate balance between complexity and generalisation. This demonstrates that the SVM was effective in finding optimal hyperplanes that maximise the distance margin between classes for the chosen problem space. Performance and F1 score of the SVM model is visualised in Figure 8 below.

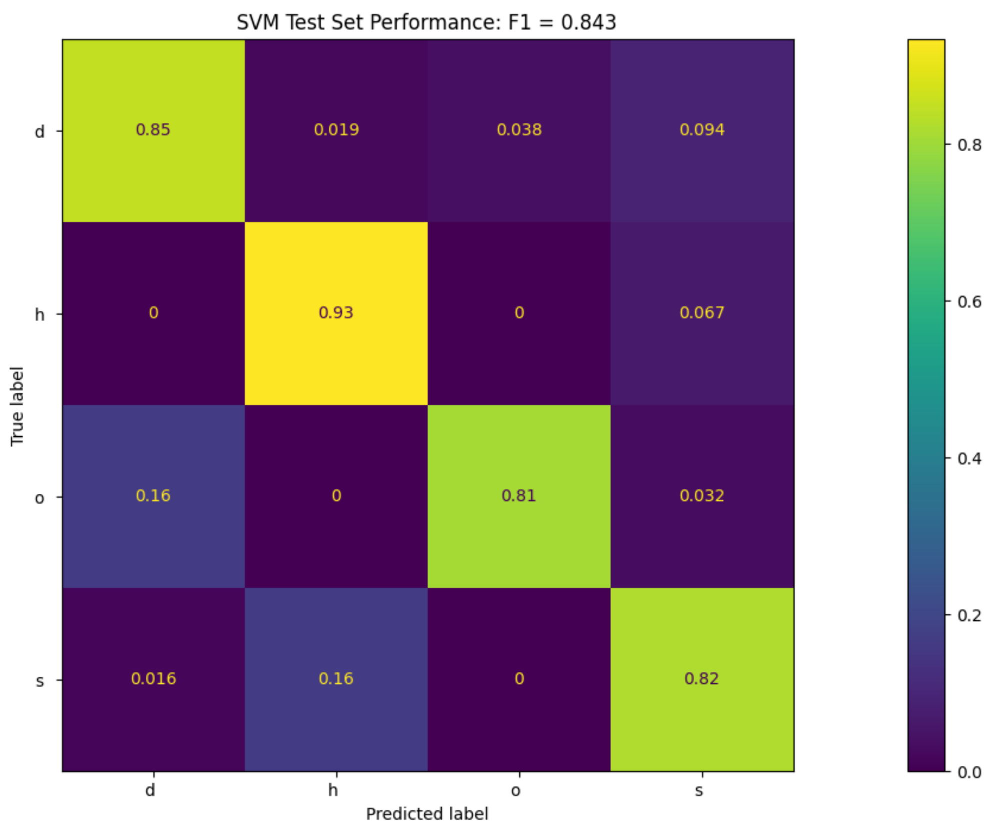


Figure 8: Confusion Matrix to Visualise the Performance of the SVM model on the Test Data Set, and to Compare its Performance to the CKNN and Random Forest Models.

The random forest model attained an F1 score of 0.794 and exhibited reasonable performance across all classes. However, the model demonstrates minor difficulties in correctly identifying class 'd'. This variance in recall may arise from the model's random nature, where certain trees may underperform if they do not include discriminative features for classes. The confusion matrix representing the random forest model's performance and F1 score is shown in Figure 9 below.

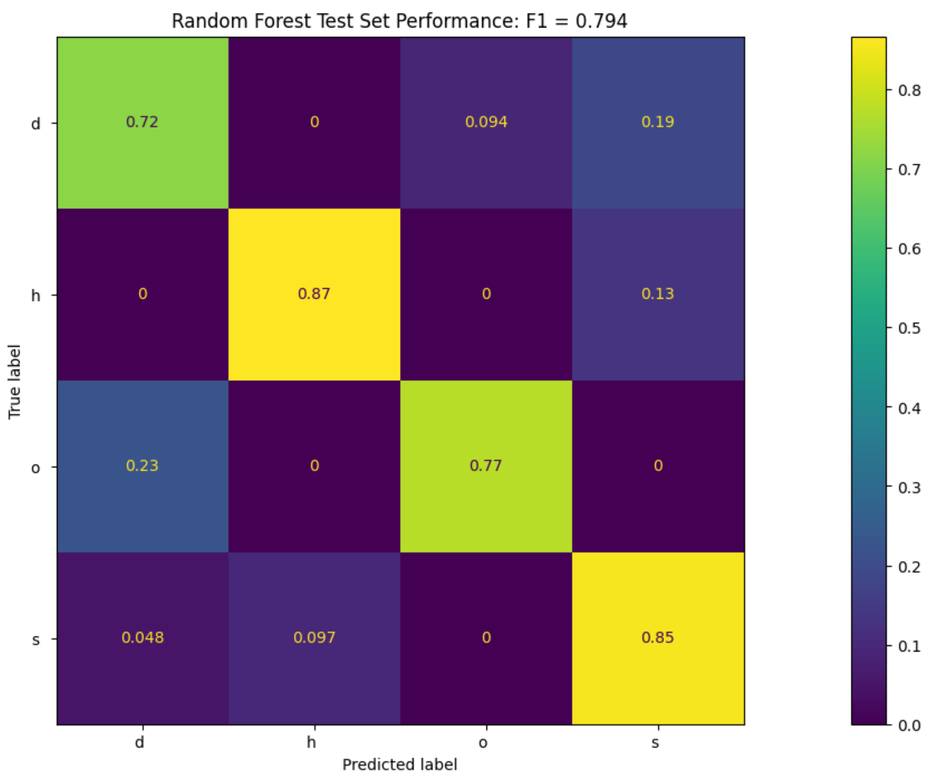


Figure 9: Confusion Matrix to Visualise the Performance of the Random Forest model on the Test Data Set, and to Compare its Performance to the CKNN and SVM Models.

The discrepancies observed in model performance may be attributed to the inherent differences in how each model calculates and handles data. All models seem to be impacted by class imbalance or sub-optimal hyperparameter refinement to an extent. The variance in recall across classes, particularly the CKNN model, highlights the challenges in accurately classifying minority classes. The SVM outperforms the CKNN and random forest models, demonstrating its ability to effectively manage a multi-class problem utilising a one-vs-one approach. However, defining the best model may depend on specific application needs, where simpler models such as CKNN or random forest may be preferred despite slightly lower performance metrics or higher computational times.

DEEP NETWORKS

1. **Network Design**

The neural network utilises a tailored VGG-like architecture. This design was selected as the dense layers lead to slightly superior performance and enhanced computational efficiency compared to a ResNet in scenarios where lower filter counts are involved in model development. The network consists of three main convolutional blocks. The first block includes two Conv2D layers with 32 filters of 7x7 kernel size. The larger kernel size in the initial block was chosen to capture broad image features early in the network.

Each convolutional layer is followed by batch normalisation to reduce computation time. Spatial Dropout2D is implemented at a rate of 20% following the first block's Conv2D layers to prevent overfitting and improve generalisation. The block ends with a MaxPooling2D layer to reduce feature map dimensionality. The size of the filters decreases to 5x5 and then to 3x3 while increasing the number of filters to 64 and then to 128 as the network progresses. This allows the network to capture increasingly detailed features necessary for accurate classification. The convolution blocks are flattened, and the vector is passed to a dense layer of 512 units, followed by another dense layer with 10 units corresponding to the number of classes. Swish and ReLU activation functions were compared during experimentation. Ultimately, Swish activation was selected for the final network due to its superior performance metrics. A complete model breakdown is shown in Table 1 of the Appendix.

Training was conducted using the Adam optimiser with a learning rate of 0.001. A batch size of 32 was implemented to balance computational efficiency with practical gradient estimation, given the total training dataset size of 1000 images. The model was set to train for up to 50 epochs with an early stopping mechanism based on validation loss and patience set to five epochs. Non-augmented training converged after 5.2 minutes at epoch 18. Augmented training converged after 10.2 minutes at epoch 41. To demonstrate convergence, training and validation loss were plotted against epochs, shown in Figure 10 and Figure 11. The plots demonstrate a consistent decline in loss values reaching a global minimum, indicating successful convergence. However, the validation loss exhibits increased fluctuations in the model trained with data augmentation relative to the non-augmented network.

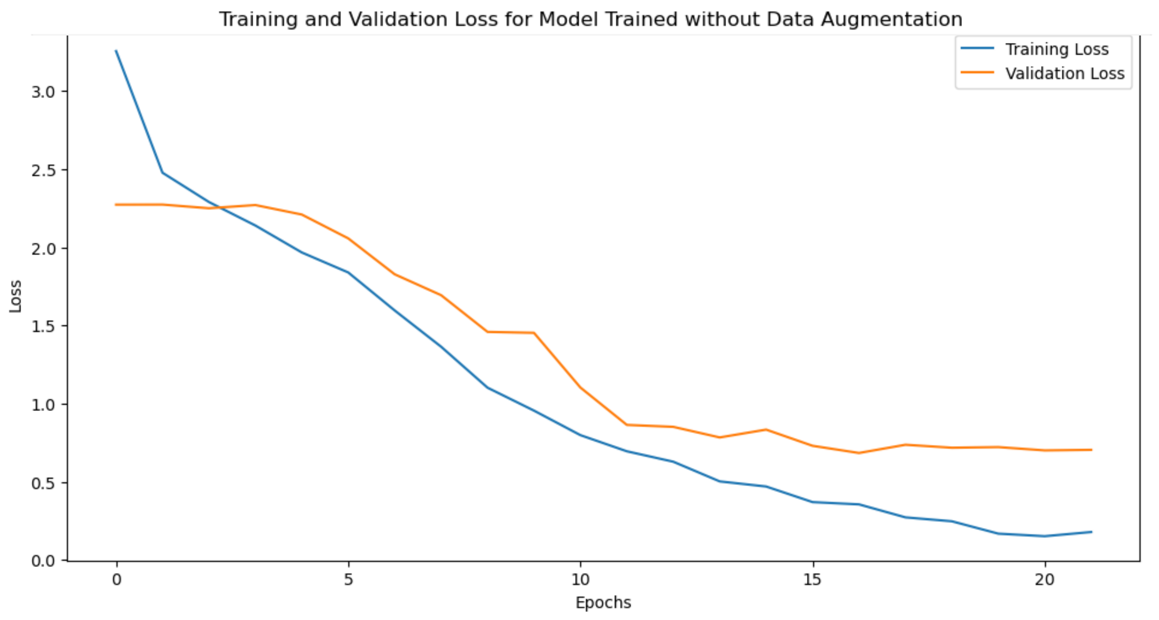


Figure 10: Epochs versus Loss to Determine Successful Convergence of the Network Trained without Augmented Data.

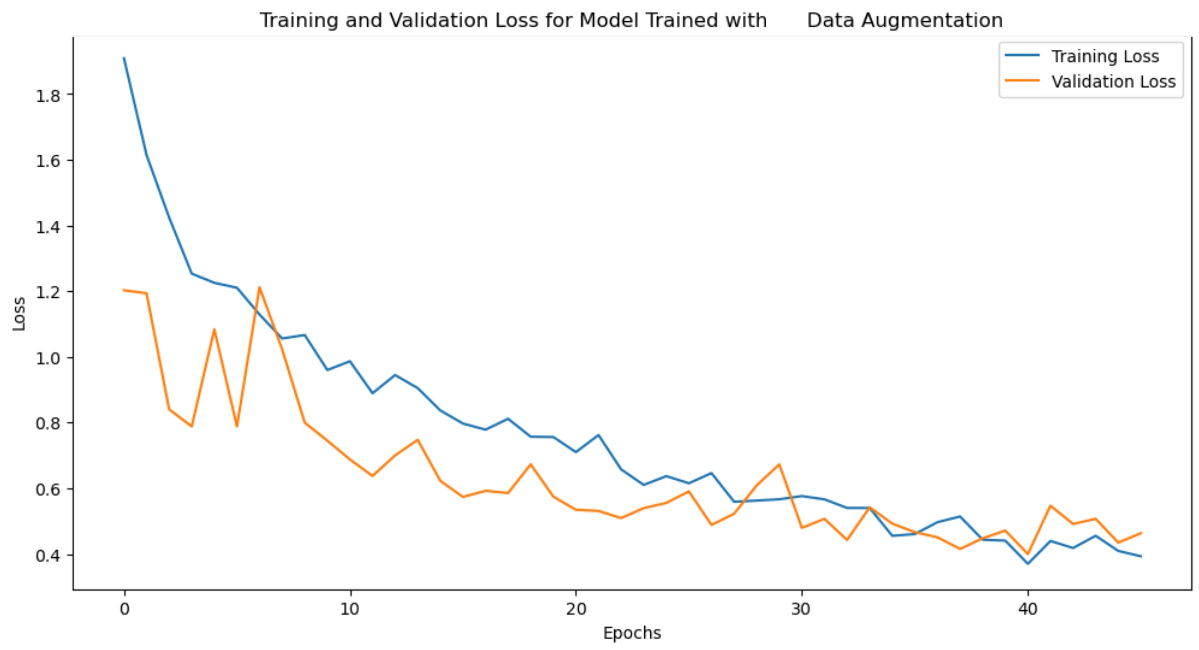


Figure 11: Epochs versus Loss to Determine Successful Convergence of the Network Trained with Augmented Data.

2. **Data Augmentation**

Given the relatively small size of the dataset, data augmentation was implemented to enhance the performance of the neural network. The data augmentation techniques applied included rotation up to 15 degrees, width and height shifts up to 10%, shear transformations up to 10%, zoom up to 10%, and horizontal flipping. Implementing these methods introduced realistic variations that may potentially occur in real-world scenarios. Augmentation also artificially expanded and increased the diversity of the training dataset without the need for additional data, attempting to promote increased model generalisation and performance.

3. **Evaluation and Analysis**

The VGG-like neural network trained without data augmentation achieved high training and testing F1 scores. However, the neural network trained on augmented data demonstrated superior performance metrics on the test data set, achieving F1 scores of 0.92 on the training set and 0.87 on the test data set. This suggests that data augmentation may contribute to increased model performance and enhance model generalisation. The enhanced range of features represented during training may also influence the performance increase. The slightly lower training F1 score on the tailored VGG network trained with data augmentation may be attributed to the model learning from an increasingly complex data set. The SVM attained a comparable training score of 0.81. However, the model scored significantly lower on the test data set. This discrepancy may result from the SVM failing to generalise to unseen data due to overfitting or the inability to handle complex image classification tasks relative to neural networks. F1 scores attained from each model are shown in Table 2 below.

Table 2: Model and F1 Score on the Training and Test Data Set for the SVM and Neural Network Trained with Augmented and Non-Augmented Data.

Model	**F1 Score (Training)**	**F1 Score (Test)**
SVM	0.87	0.35
Neural Network Trained on Non-augmented Data	0.95	0.81
Neural Network Trained on Augmented Data	0.92	0.87

Training times for the neural networks were significantly slower than the SVM. Additionally, the network trained on augmented data exhibits the most extended processing times, potentially due to the computational demands of handling augmented images that expand the dataset's variability and complexity. Inference times depict similar trends, with the SVM showcasing considerably faster performance metrics. The slower inference and training times for the DCNNs may be attributed to the deeper architecture of the neural network and the increased computational demand of processing images through each layer. However, the minimal difference in inference times between augmented and non-augmented models indicate that once the model is trained, the inference times for a given network may stabilise. This consistency may be attributed to the inference process involving a forward pass through the network using learned weights, without needing additional adjustments or learning. Training and inference times seem to scale with the size of the data set. Results are displayed in Table 3 below.

Table 3: Training Time and Inference Times for the SVM and Neural Network Trained with Augmented and Non-Augmented Data.

Model	**Training Time, seconds (s)**	**Inference Time: Training Set, seconds (s)**	**Inference Time: Test Set, seconds (s)**
SVM	1.58	0.53	5.40
Neural Network Trained on Non-augmented Data	312	1.11	9.53
Neural Network Trained on Augmented Data	612	0.92	9.83

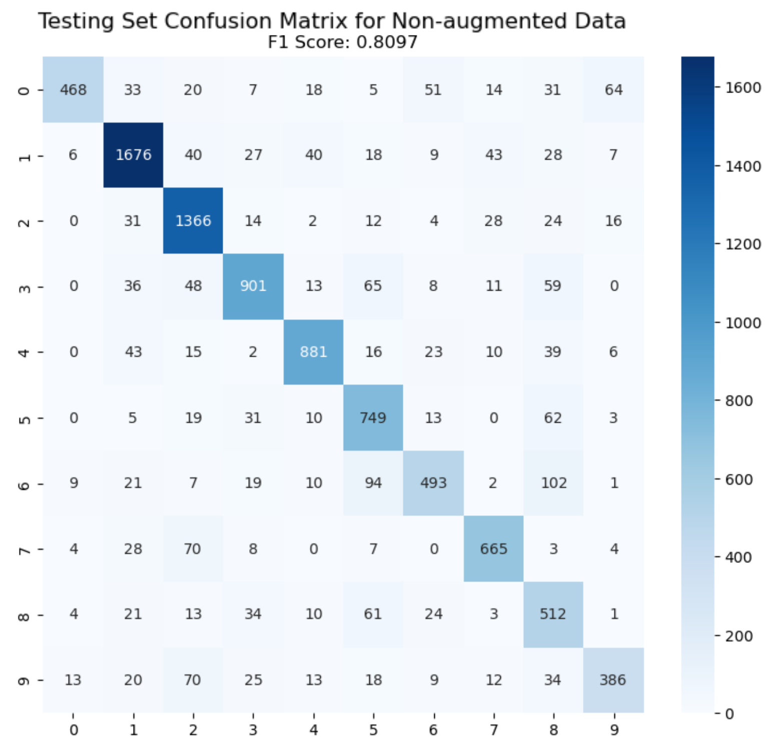
Minority class predictions seem to improve with the DCNNs and further improve when augmented data is utilised. This improvement may be attributed to the augmented model's exposure to increasingly varied samples during training, which includes better representation of minority classes through additional data variations. Confusion matrices of the SVM and DCNN's are outlined under Figure 1, Figure 2 and Figure 3 in the Appendix. To conclude, the SVM offers increasingly efficient training and inference times. However, its performance is significantly reduced relative to a DCNN. This suggests this algorithm type is unsuitable for increasingly complex class recognition tasks. The DCNN trained with augmented data offers the best balance between performance and generalisation. This potentially makes this model the most suitable for scenarios where training or inference times are not paramount and where accurate classification is required across diverse or imbalanced datasets.

APPENDIX

Table 1: Model Architecture and Detailed Parameter values of the Network Developed for the Classification Task.

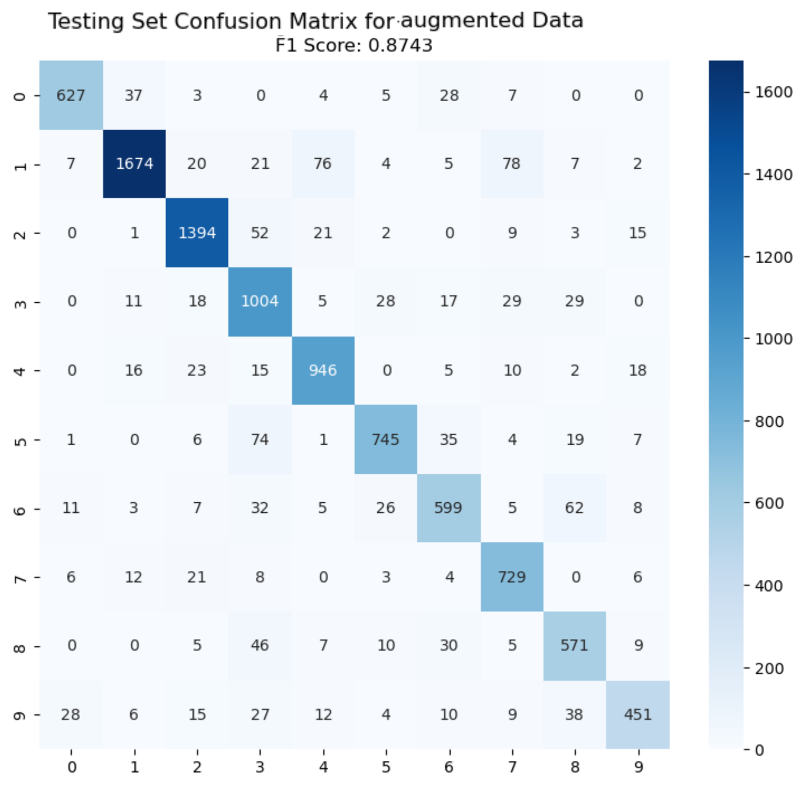
Layer (type)	**Output Shape**	**Param #**
img (InputLayer)	(None, 32, 32, 3)	0
conv2d (Conv2D)	(None, 32, 32, 32)	4,736
batch_normalization (BatchNormalization)	(None, 32, 32, 32)	128
conv2d_1 (Conv2D)	(None, 32, 32, 32)	50,208
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 32)	128
spatial_dropout2d (SpatialDropout2D)	(None, 32, 32, 32)	0
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	51,264
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 64)	256

```
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_3 (Conv2D)               │ (None, 16, 16, 64)     │       102,464 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_3           │ (None, 16, 16, 64)     │           256 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ spatial_dropout2d_1             │ (None, 16, 16, 64)     │             0 │
│ (SpatialDropout2D)              │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 8, 8, 64)       │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_4 (Conv2D)               │ (None, 8, 8, 128)      │        73,856 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_4           │ (None, 8, 8, 128)      │           512 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_5 (Conv2D)               │ (None, 8, 8, 128)      │       147,584 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_5           │ (None, 8, 8, 128)      │           512 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ spatial_dropout2d_2             │ (None, 8, 8, 128)      │             0 │
│ (SpatialDropout2D)              │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 2, 2, 128)      │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 512)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 512)            │       262,656 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 10)             │         5,130 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 699,690 (2.67 MB)
 Trainable params: 698,794 (2.67 MB)
 Non-trainable params: 896 (3.50 KB)
```



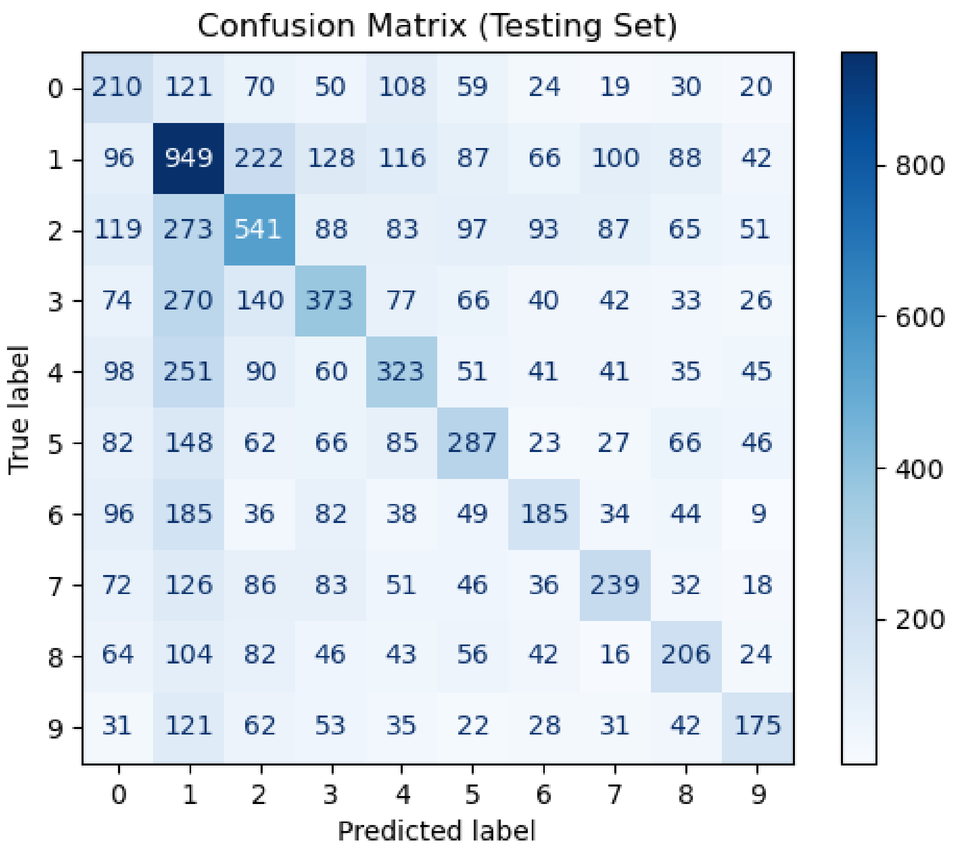
	0	1	2	3	4	5	6	7	8	9
0	468	33	20	7	18	5	51	14	31	64
1	6	1676	40	27	40	18	9	43	28	7
2	0	31	1366	14	2	12	4	28	24	16
3	0	36	48	901	13	65	8	11	59	0
4	0	43	15	2	881	16	23	10	39	6
5	0	5	19	31	10	749	13	0	62	3
6	9	21	7	19	10	94	493	2	102	1
7	4	28	70	8	0	7	0	665	3	4
8	4	21	13	34	10	61	24	3	512	1
9	13	20	70	25	13	18	9	12	34	386

Figure 1: Confusion Matrix Representing the DCNNs Performance on the Test Set Utilising Non-Augmented Data



	0	1	2	3	4	5	6	7	8	9
0	627	37	3	0	4	5	28	7	0	0
1	7	1674	20	21	76	4	5	78	7	2
2	0	1	1394	52	21	2	0	9	3	15
3	0	11	18	1004	5	28	17	29	29	0
4	0	16	23	15	946	0	5	10	2	18
5	1	0	6	74	1	745	35	4	19	7
6	11	3	7	32	5	26	599	5	62	8
7	6	12	21	8	0	3	4	729	0	6
8	0	0	5	46	7	10	30	5	571	9
9	28	6	15	27	12	4	10	9	38	451

Figure 2: Confusion Matrix Representing the DCNNs Performance on the Test Set Utilising Augmented Data



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	210	121	70	50	108	59	24	19	30	20
1	96	949	222	128	116	87	66	100	88	42
2	119	273	541	88	83	97	93	87	65	51
3	74	270	140	373	77	66	40	42	33	26
4	98	251	90	60	323	51	41	41	35	45
5	82	148	62	66	85	287	23	27	66	46
6	96	185	36	82	38	49	185	34	44	9
7	72	126	86	83	51	46	36	239	32	18
8	64	104	82	46	43	56	42	16	206	24
9	31	121	62	53	35	22	28	31	42	175

Figure 3: Confusion Matrix Representing the SVMs Performance on the Test Set