Question 1: Regression

1.1 Data Pre-Processing

The dataset comes pre-segmented into distinct training, validation and testing sets, each comprising 100 columns and varying length, depending on purpose. Additionally, the data set comes normalized through an unsupervised, equal-interval binning method. As a consequence, some standard pre-processing measures such as standardisation are unnecessary. Other minor pre-processing interventions are seen as unnecessary and potentially risky, due to underlying size-complexity of the dataset.

1.2 Key Parameters

Noting that the dependent variable of 'ViolentCrimesPerPop' is consistent, each model will be reviewed individually. With the statsmodels.api package, the process of selecting linear model parameters and formular is automatic. Ridge & Lasso regression are functionally similar in this context. However, selecting the appropriate λ is manual and extremely important. With prediction strength being tested explicitly, RMSE is arguably the most important metric, so selecting λ will be done against λ through the validation dataset, testing λ values from 1 down to 10^{-8} within the respective model, comparing the resultant RMSE with the λ.

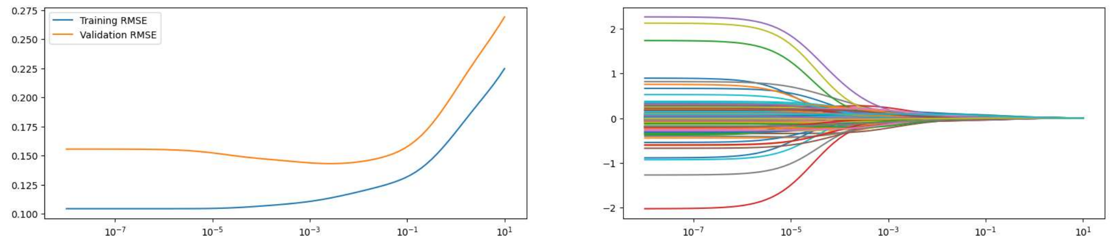


Figure 1: Visualisation of λ's effect on RMSE (Left) and coefficients (Right) with **Ridge Reg**. On the right, most coefficients appear to have converged at ~10^{-3} or well before, indicating a high level of redundant/weak coefficients.

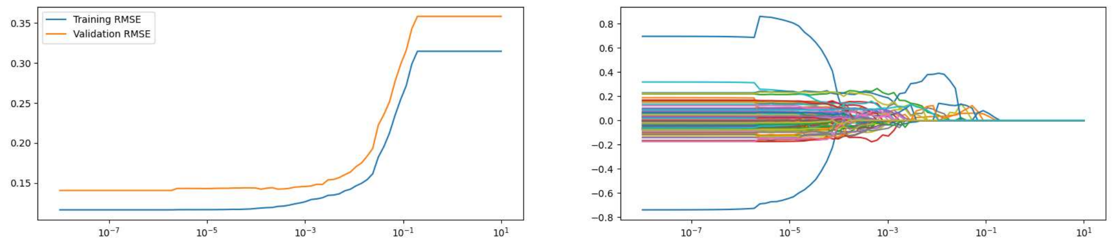


Figure 2: Visualisation of λ's effect on RMSE (Left) and coefficients (Right) with **Lasso Reg**. Conversion appears to occur at roughly the same time as Ridge, but there is significantly more volatility, possibly due to Lasso's more aggressive regularisation.

On the left, Figure 1 & 2 both demonstrate that -roughly- as λ increases along X, RMSE decreases and coefficients move towards 0. Between both Figures there are two λ values to select. Ridge's Validation RMSE of $\sim 2.9 \times 10^{-3}$ was selected for λ, as there is a necessity for a stronger regularisation to suppress the multitude of weak coefficients identified in both Figures.

1.3 Evaluation

Immediate regression results for the Linear Model obtained from r2_score of 0.890 indicates a good fit. Figure 3 explores model validity through an analysis of assumptions, while Figure 4 explores model accuracy & predictive power. With these plots, it can be observed:

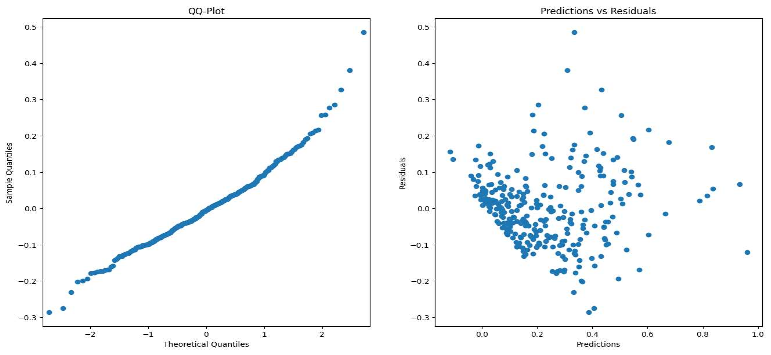


Figure 3: Visualisations for **linear model** assumption tests using the plot_diagnoistics function from CAB420_Regression_Example_2_Regularised_Regression.ipynb. There is a QQPlot on the left and a Pred vs Resid on the right.

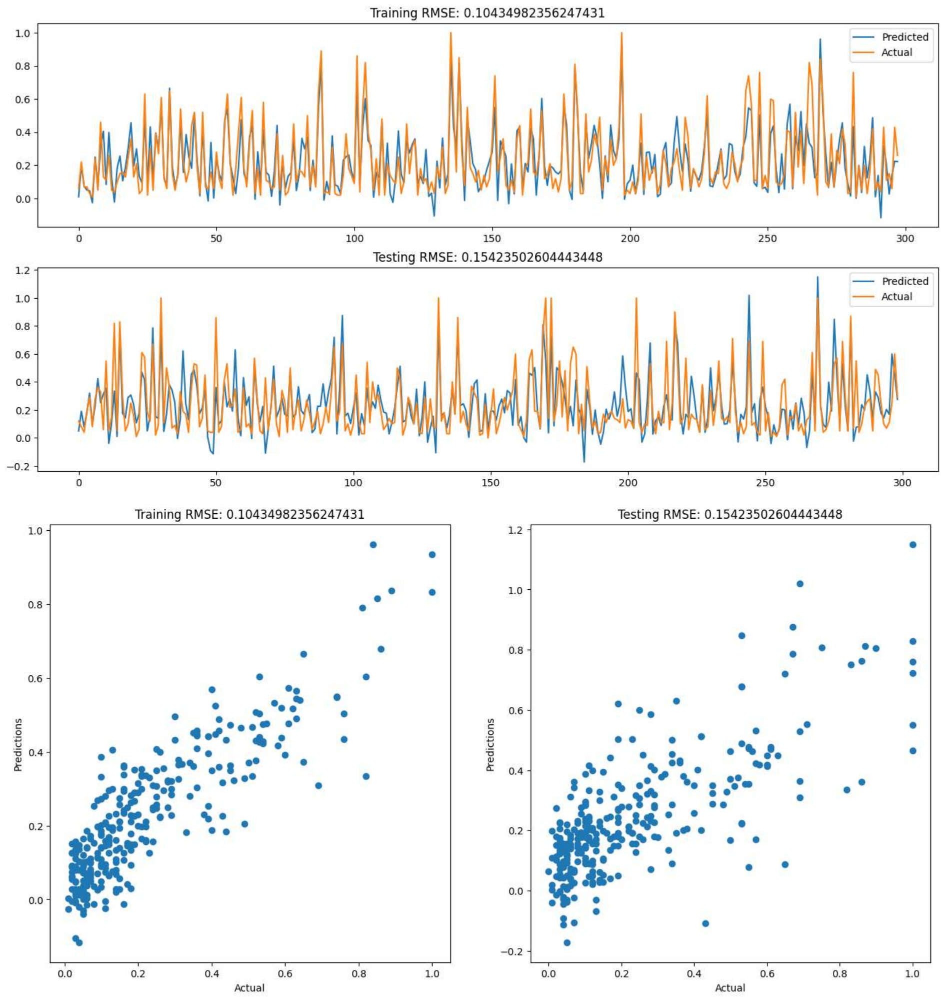


Figure 4: Two visual representations for the **linear model's** predictive power. Above each is the set's respective RSME value. Using the plot_diagnoistics function from CAB420_Regression_Example_2_Regularised_Regression.ipynb.

- There is evidence of a normal distribution, as most points align with such a trend. Though, with notable outlier tails following an increasing curve.
- The spread of residuals decreasing with x indicate heteroscedasticity within the model.
- There is little evidence to suggest that the linear model has been overfit, and the RSME value for testing indicates a very high level of predictive power. It seems to capture the major trends in the data, though it often overestimates peaks.
- Predictions and ground truth do not indicate much if any level of correlation.

Moving along to Ridge regression, its R2 result of ~0.758 indicates a decrease in model validity from linear regression, indicating that Ridge might be overly penalising coefficients. Figure 5 and 6 use identical functions to explore model validity and predictive power. In them it can be observed:

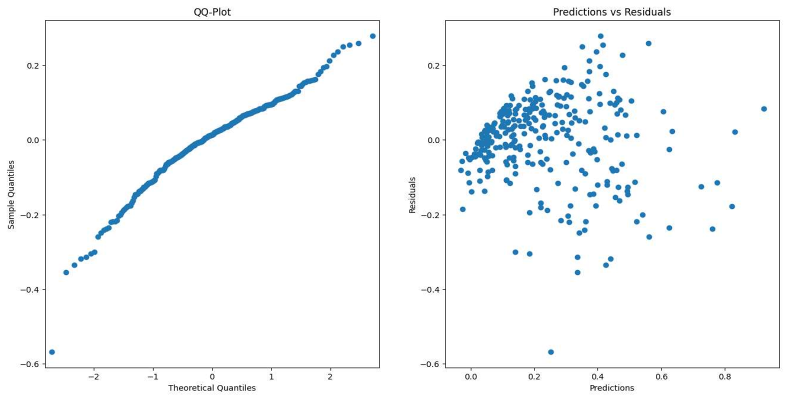


Figure 5: An output of the plot_diagnoistics function using a **Ridge regression** model.

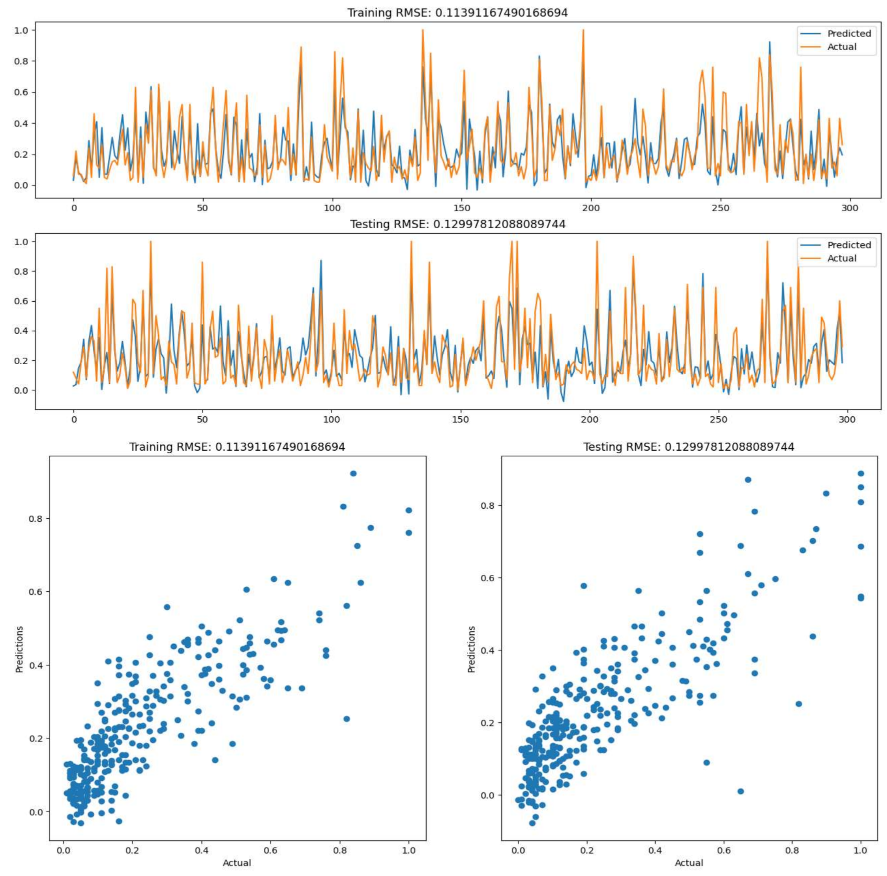


Figure 6: Two different visual representations for the **Ridge regression** model's predictive power, using the plot_diagnoistics function.

- Compared to Figure 3, the QQPlot arguably indicates greater evidence of normally distributed values.
- There is still a clear pattern indicating heteroscedasticity within the model, however it has been effectively mirrored from Figure 3.
- Figure 6 indicates a general improvement for predictive power for testing set over Figure 4.
- There are frequent underestimations of peaks in prediction over actual.
- There is some visual improvement with model correlation.

Finishing with Lasso Regression, its R2 score of 0.701 indicates an even worse fit from Ridge regression, potentially suggesting the dataset having a more linear nature. Figure 7 is an exploration of model validity through identical methods to above figures, through them it can be observed:

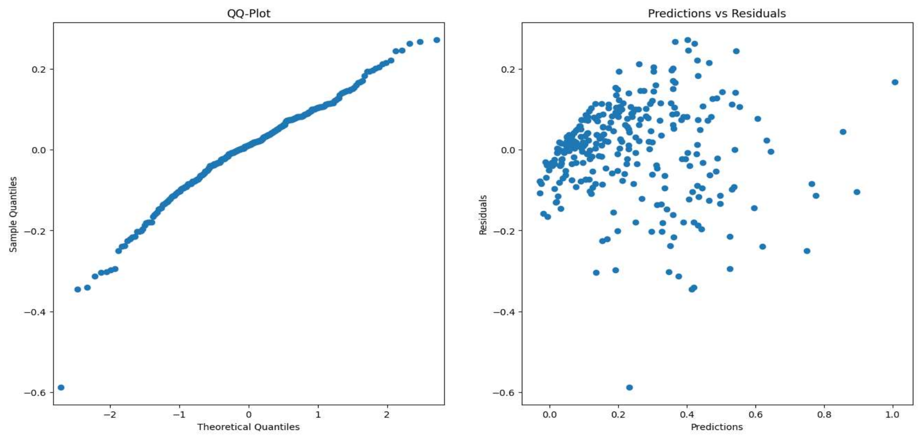


Figure 7: An output of the plot_diagnoistics function using a **Lasso Regression**.

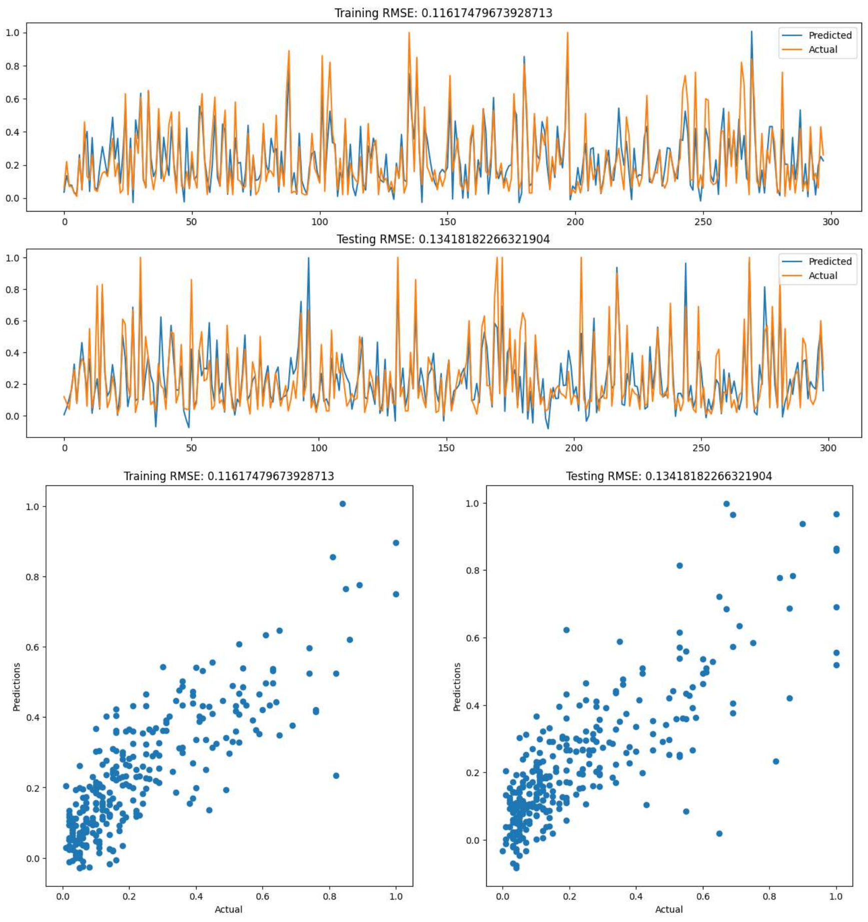


Figure 8: Two different visual representations for the **Lasso regression** model's predictive power, using the plot_diagnoistics function.

- Compared to Figure 5, the QQ-plot does not visually improve upon the assumption of a normal distribution.
- There is still a pattern indicating heteroscedasticity within the model consistent to Figure 5.
- Figure 8 indicates a general improvement for predictive power for the testing set and marginal decrease for training over Figure 4, though a decrease over Figure 6.

The overall impression of all three models is a consistent thread of heteroscedasticity, likely resultant of the number of redundant parameters within the models, as addressed in Figures 1 & 2 (right) plots. Predictive power has been sacrificed for weaker R2 scores which indicate a general decrease in validity between Ridge/Lasso and Linear regression models, which is a concern considering the dataset's nature.

1.4 Socio-Economic Considerations

The socio-economic nature of the dataset leaves certain ethical concerns when discussing model accuracy and validity between all three results. The violation of the assumption of homoscedasticity in all three models may lead to incorrect statistical inferences. The possibility that these models could be used to make certain funding or policy decisions may lead to certain incorrect predictions or assumptions.

Question 2: Classification

2.1 Data Pre-Processing

The Dataset used is comprised of aerial sensor data capturing different spectral reflectance properties of ground beneth. Given the task of training several multi-class classifiers for classifying land type using spectral data, with limited data available (523 samples total) cropping cannot be conducted in spite of general class & spectral data imbalance as seen in Figure 9 & 10.

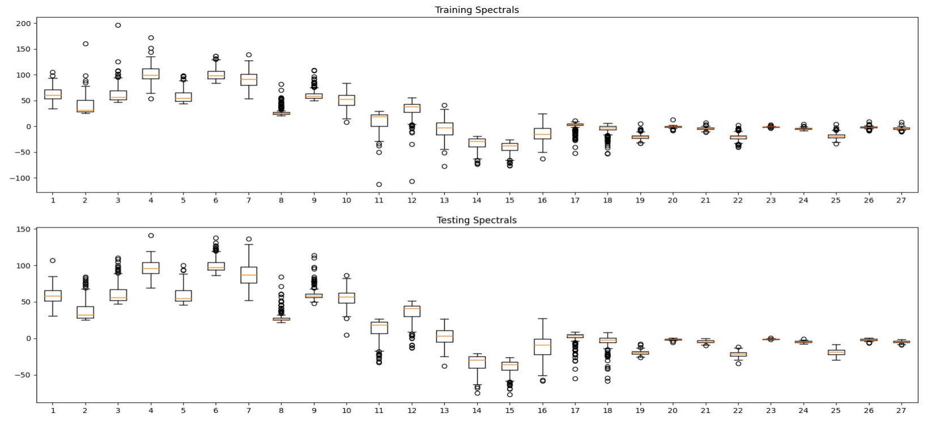


Figure 9: Boxplot of training & testing 'X' sets to visualise spectral data irregularity/noise.

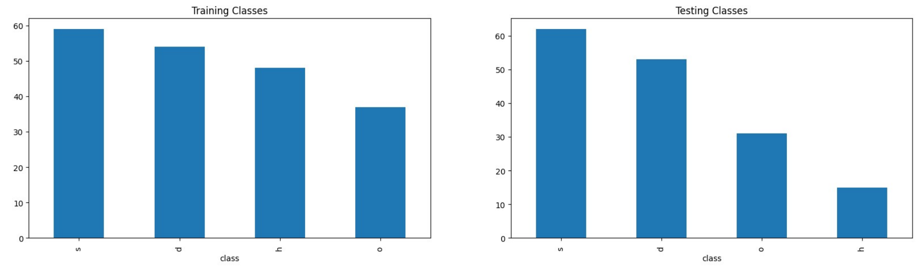


Figure 10: Bar plot of class counts imbalance within testing and training sets.

While standardisation is generally good practice for classification performance, as it makes it easier for the model to consider all dimensions equally, this would be harmful overall. There is no use in a classifier optimised for balance if the reality is that the dataset is just as imbalanced.

2.2 Hyper-Parameter Selection

Nested loops were determined to be the preferred grid-search method. A K-Nearest Neighbours classifier, Random Forest and an ensemble of Support Vector Machines will all be tested in this fashion and given brief discussion over their specific hyperparameters, with the best result being the combination of hyperparameters resulting in the weighted f1 score closest to 1.

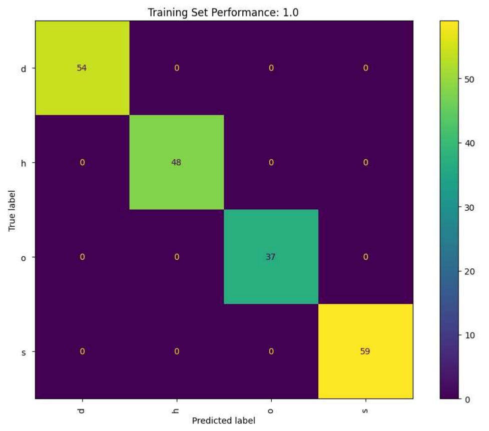
For K-Nearest Neightbours (CKNN), the hyperparameter 'n_neighbors' determines the number of neighbours to use in queries. Generally, at =1 CKNN tends to closely follow training data, which often results in a high training score, but weaker testing score. Weights can either be uniform or distance and determines if points are weighted equally or by their inverse. Metric is either cityblock or euclidean (which is just minkowski at p = 2). Grid search of the validation set determined the best combination was {'k': 1, 'weight': 'uniform', 'metric': 'euclidean', 'f1': 0.8702687905183208}.

For Random Forest (RF), the hyperparameter n_estimators determines the number of trees within the forest. Higher values *can* equate to higher accuracy estimations. Max depth determines how deep a tree can go from 1 → ∞. By passing None, nodes are expanded until all leaves are pure. Class weight determines how classes are weighed. Passing None will ensure all classes will have weight one. Balanced mode uses the values of y to automatically adjust weights inversely proportional to class frequencies. Balanced subsample is the same, except that weights are computed on the bootstrap sample for every tree growth. Gridsearch determined the best combination of parameters as {'depth': None, 'n_estimators': 15, 'class weight': 'balanced_subsample', 'f1': 0.8481584525265444}

Within SVM's, C determines the regularisation parameter and must be strictly positive. It roughly determines how much the optimiser should avoid misclassifying each example. Smaller values allow for more misclassifications in the training data, which can result in a wider margin between classes / vice versa. Kernel is limited to 'linear', 'poly', 'rbf' and 'sigmoid' and specifies the kernel type used in the algorithm. They are more relevant to certain mutually exclusive parameters, specifically degree and gamma, which determine the degree of the 'poly' kernel function and the kernel coefficient for 'rbf', 'poly' and 'sigmoid' respectively. The best combination determined by grid search was {'C': 1000.0, 'kernal': 'rbf', 'gamma': 'scale', 'f1': 0.8601532465606897}.

2.3 Evaluation

Confusion charts for the training and testing set for K-Nearest Neighbours classifier (CKNN) are shown in Figure 11. Weighted F1 scores of 1.0 and 0.857 are obtained for the sets. Misclassification is occurring generally between 'similar' classes (such as 6 'Sugi' (s) forests being identified as 'Hinoki' (h)). There is also the possibility of class imbalance effecting performance between 'Other' (o) / 'Sugi' (s) forests, though it is also possible that statistically the two are similar.



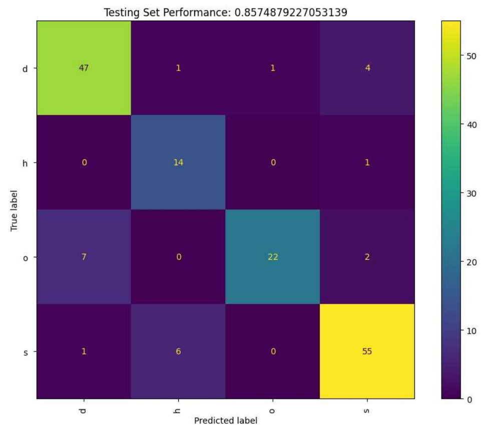


Figure 11: CKNN Confusion charts for training (left) and testing (right) sets.

Confusion charts for Random Forest (RF) training and testing sets are shown in Figure 12. Weighted F1 scores of 1.0 and 0.804 are obtained for the sets, observing a mild numerical decrease in model performance, but a somewhat significant increase in misclassifications in the testing set from Figure 11. Specifically between 'Sugi' (s) and 'Hinoki' (h) forests and between 'Mixed deciduous' (d) and 'Other' (o) / 'Sugi' (s) forests. These are consistent to misclassifications in Figure 11, but the prevalence suggests that RF may potentially be less suitable for classification with the aerial sensor data.

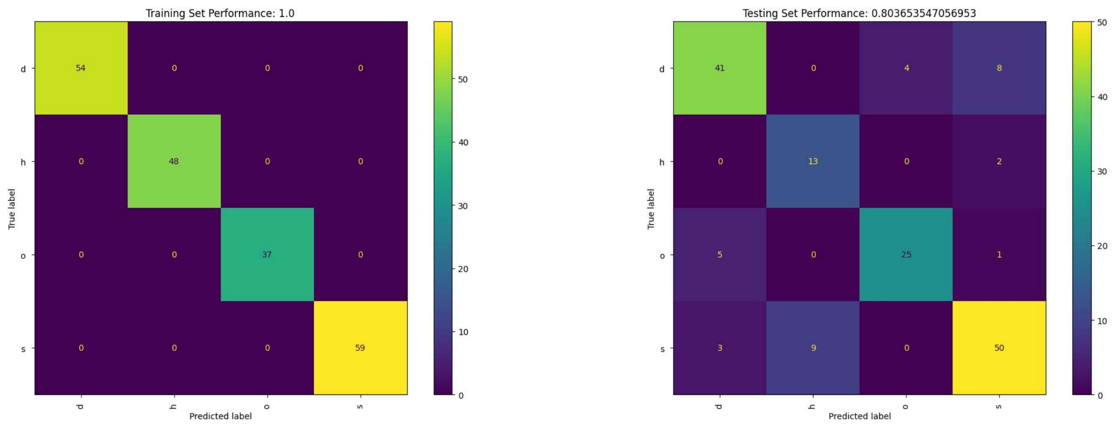


Figure 12: RF Confusion charts for training (left) and testing (right) sets.

Confusion charts for Support Vector Machines (SVM) training and testing sets are shown in Figure 13. Weighted F1 scores of 1.0 and 0.849 are obtained for the sets, observing a inconsequential numerical decrease in model performance from Figure 11, but an overall increase from Figure 12. A minor improvement with misclassifications can also be observed, in comparison to Figure 13, but overall still worse than Figure 11. With all three comparisons, it can be observed that:

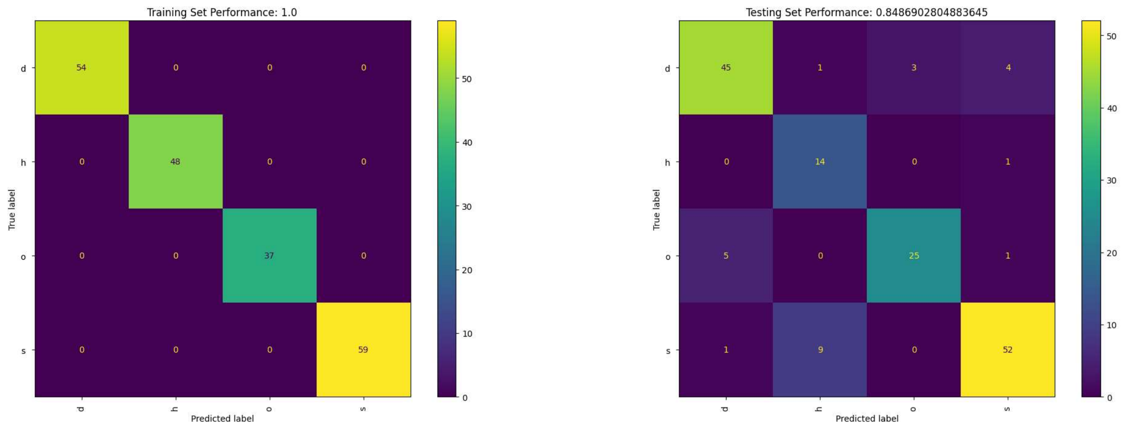


Figure 13: SVM Confusion charts for training (left) and testing (right) sets.

- While there is some evidence for class imbalance affecting model performance occurring in all three models, it is not overt. Suggesting that certain pre-processing decisions were justified despite general practice.
- Confusion between statistically similar classes is relatively consistent between all three models but is more prevalent in RF & SVM testing sets. Which might be resultant from inconsistencies with grid search determined parameters.

Overall, all three models performed exceptionally, with uniformly perfect training set results and relatively high testing set performance. Class imbalance does not appear to affect model performance significantly, if at all, evidenced by frequent misclassification of similar data, rather than a bias against larger data. Which suggests that standardisation may be unnecessary, or harmful.

Question 3: Training and Adapting Deep Networks

3.1 General Neural Network Design

One model needs to be developed from scratch for classifying street-view house number images. A Resnet network is desirable for this task, given the general size of the testing set (10,000 images) necessitating high performance and capability in handling deep networks. Specifically, the resnet_v1 function from the CAB420_DCNNs_Example_5_ResNet.ipynb example is used, with 1 residual stage with 16 and 32 filters and a single residual block. While given 200 epochs, a callback exists to catch the network after 5 epochs with minimal change in loss (consolidation). This can be seen in Figure 15. The network uses a batch size of 64 with an Adam optimiser on a learning rate $(\lambda = 3e - 3)$ used to offset expected overfitting from current settings. Appendix A shows diagram of network.

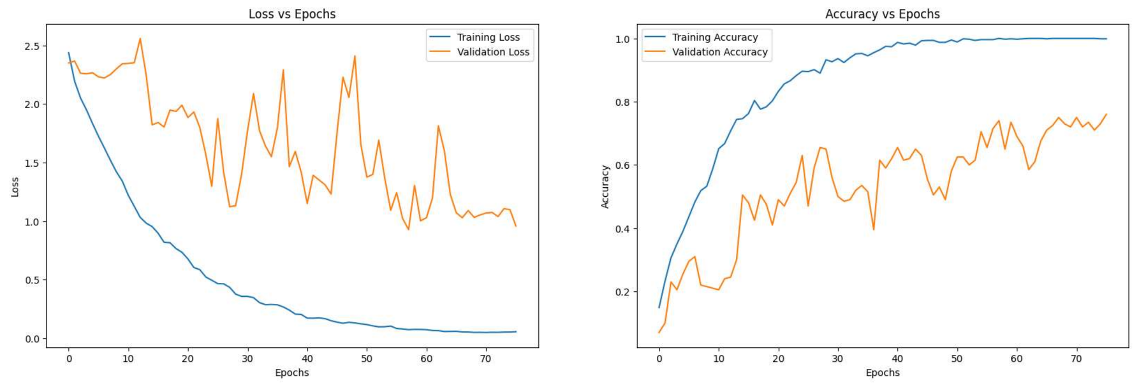


Figure 15: Model training performance, showing loss (left) and accuracy (right).

The loss trend is generally decreasing, but scattered peaks indicate overfitting. This suggests the network is learning and improving, with episodes of overfitting during training.

3.2 Data Augmentation

With the general network design completed, an identical second network was implemented with an input layer applying data augmentation as seen in Figure 16 for project specifications. Considering the nature of the dataset, augmentations must be restrained in order to prevent misclassification or data warping. Small random rotations (±2.5%), random zooms (±2.5%) and small translations (±2.5% in horizontal and vertical directions) are used.

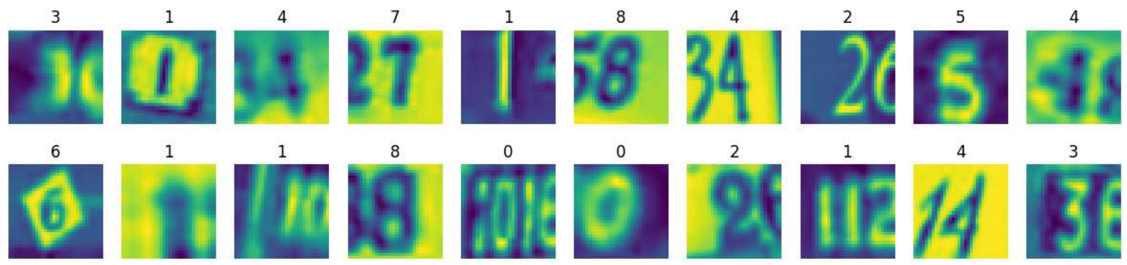


Figure 16: Data Augmentation

It is worth noting separately that while image reductions can be effective for improving network performance, it would have the opposite within current dataset. Note that in Figure 16, a reduction of even 4 x 4 pixels would affect the visible value of some images. With the modified input layer, the network was retrained with identical parameters to the previous network.

This does not mean that the network will be identical to Appendix A. Doing so would result in worse performance, as the network would be trained for an unmodified input layer. Training the network from scratch will provide the best circumstances to improve network performance.

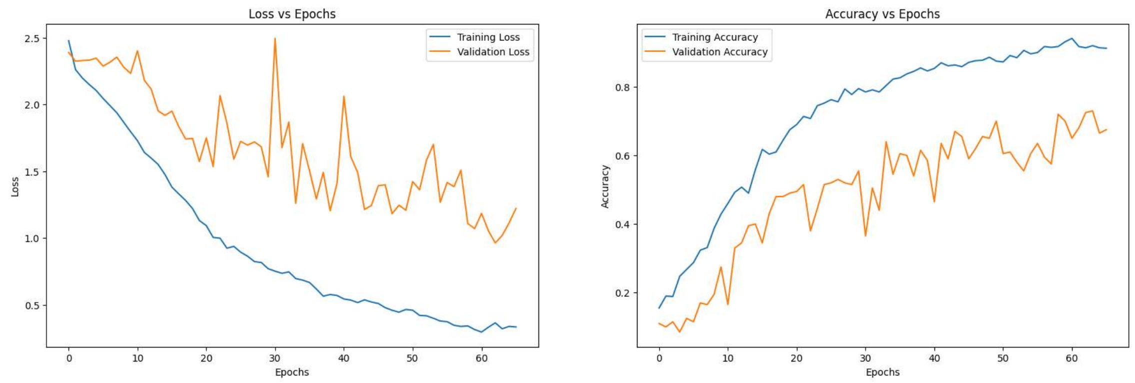


Figure 17: Loss & Accuracy vs Epochs chart w/ augmented input layer ResNet V1 model.

Figure 17's results being generally similar to the ResNet V1 network without data augmentation is unsurprising. Overfitting can be observed throughout, albeit it is unclear whether it truly converges. Appendix B shows a diagram of this network with the modified input layer.

3.3 Evaluation

In addition to the two ResNet V1 networks, an additional SVM has been trained and can be evaluated for comparison in Figure 18. Macro F1 scores of ~0.678 and 0.279 were obtained for training and testing sets. Overfitting can be observed through the drastically smaller testing F1 over training. Even without any prior dataset analysis, there is clear evidence of class imbalance within the dataset, and misclassification is high against the most populous classes. Out of all three models, the SVM trained (0.61 seconds) and compiled the fastest (~1.8 seconds).

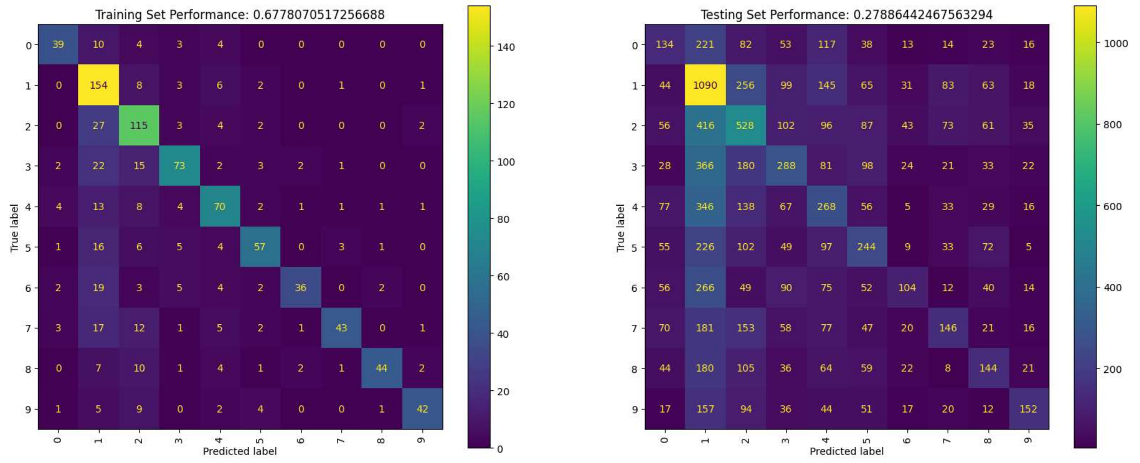


Figure 18: Confusion chart for SVM network with training (left) and testing (right) sets.

Figures 19 and 20 show confusion charts evaluating network performance with the ResNet V1 network without (Figure 19) and with (Figure 20) data augmentation applied to input layer.

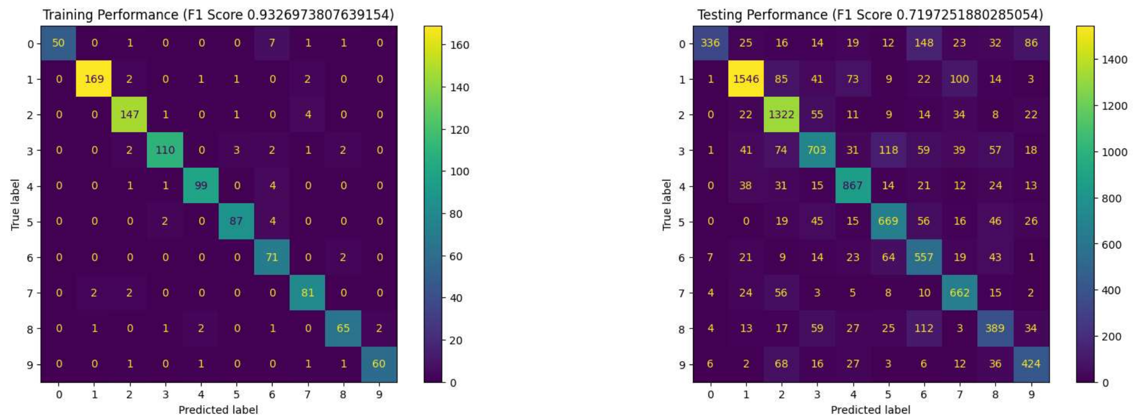


Figure 19: Confusion Chart of ResNet_V1 model w/ F1 scores, training performance on the left and testing performance on the right.

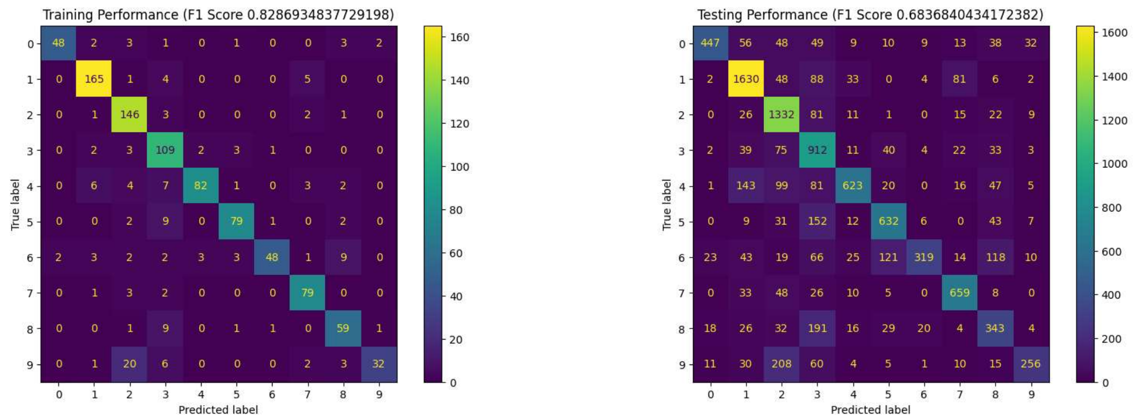


Figure 20: Confusion Chart of augmented input layer ResNet V1 model w/ F1 scores, training performance on the left and testing performance on the right.

Training and testing F1 scores of ~0.93 / 0.72 and 0.83 / 0.68 indicate a massive performance boost from the SVM network (+0.45 in testing alone). This can be reinforced visually, as neither Figures 19 or 20 display any evidence of misclassification due to class imbalance. However, there are signs of overfitting as seen in Figures 15 & 17, as similar to the SVM network, the macro F1 values for testing are notably below the training set. In addition, compilation and training times for both models are substantially slower than in SVM (~32 seconds each). Though there is no notable difference between the network with and without the augmented input layer.

Overall, the two ResNet V1 networks perform significantly better than the SVM network in terms of F1 scores and misclassification, even with SVM's vastly faster overall speed. While specified in task brief, it cannot be stated with any certainty that data augmentation was necessary. Comparing testing set performance, F1 scores and confusion chart misclassification puts both ResNet V1 networks roughly equal. However, data augmentation within this application's case is dangerous considering the possibility of changing the image's visual value. Factoring the size of the used training and testing sets, this is why data augmentation cannot be truly considered necessary.

A Network Diagram

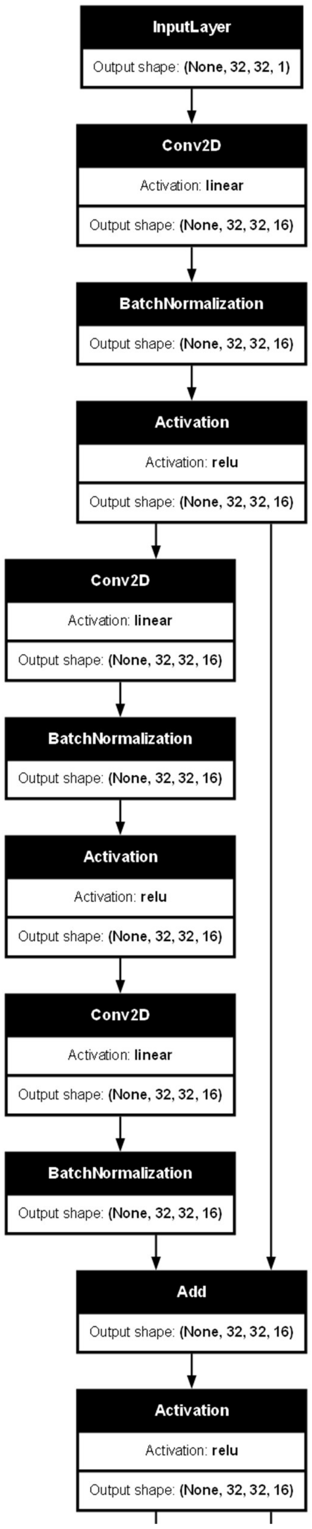

InputLayer
Output shape: (None, 32, 32, 1)
Conv2D
Activation: linear
Output shape: (None, 32, 32, 16)
BatchNormalization
Output shape: (None, 32, 32, 16)
Activation
Activation: relu
Output shape: (None, 32, 32, 16)
Conv2D
Activation: linear
Output shape: (None, 32, 32, 16)
BatchNormalization
Output shape: (None, 32, 32, 16)
Activation
Activation: relu
Output shape: (None, 32, 32, 16)
Conv2D
Activation: linear
Output shape: (None, 32, 32, 16)
BatchNormalization
Output shape: (None, 32, 32, 16)
Add
Output shape: (None, 32, 32, 16)
Activation
Activation: relu
Output shape: (None, 32, 32, 16)

Figure 21: Model used w/ unmodified input layer. Note that output has been truncated and continues on the next page.

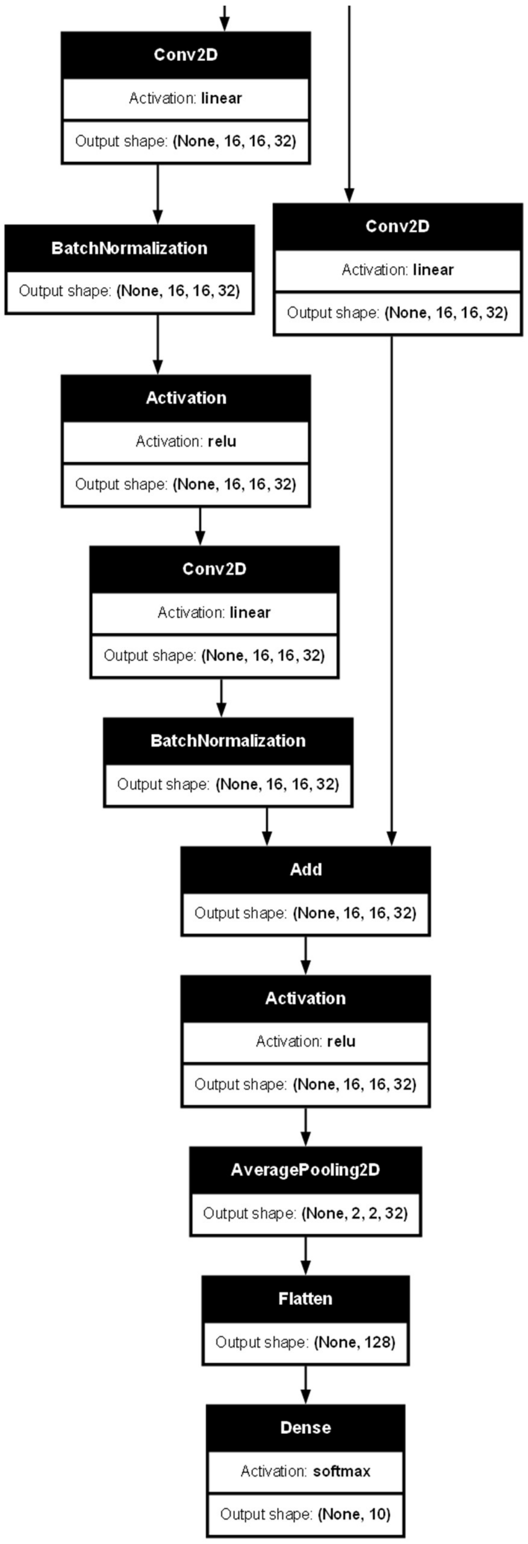

Conv2D
Activation: linear
Output shape: (None, 16, 16, 32)
BatchNormalization
Output shape: (None, 16, 16, 32)
Conv2D
Activation: linear
Output shape: (None, 16, 16, 32)
Activation
Activation: relu
Output shape: (None, 16, 16, 32)
Conv2D
Activation: linear
Output shape: (None, 16, 16, 32)
BatchNormalization
Output shape: (None, 16, 16, 32)
Add
Output shape: (None, 16, 16, 32)
Activation
Activation: relu
Output shape: (None, 16, 16, 32)
AveragePooling2D
Output shape: (None, 2, 2, 32)
Flatten
Output shape: (None, 128)
Dense
Activation: softmax
Output shape: (None, 10)

Figure 22: Model w/ unmodified input layer used in the solution. Diagram continues from previous page.

B Network Diagram w/ Modified Input Layer

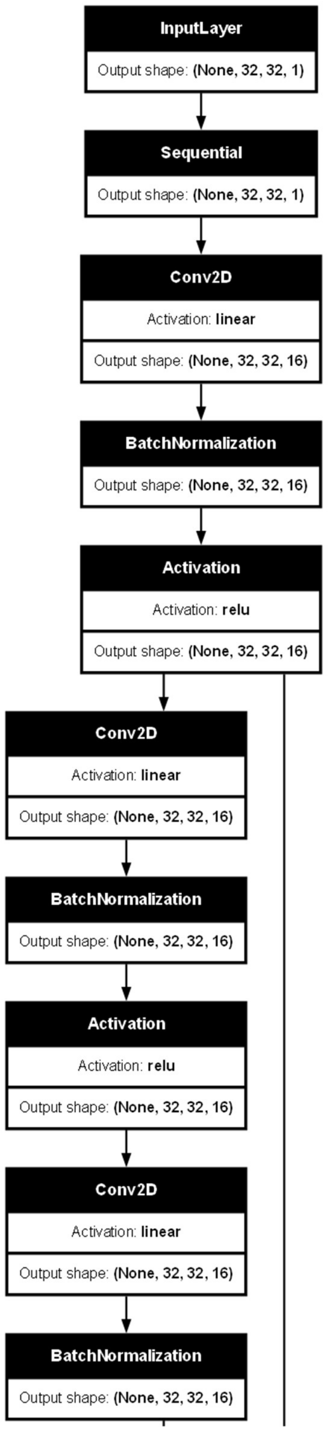

InputLayer
Output shape: (None, 32, 32, 1)
Sequential
Output shape: (None, 32, 32, 1)
Conv2D
Activation: linear
Output shape: (None, 32, 32, 16)
BatchNormalization
Output shape: (None, 32, 32, 16)
Activation
Activation: relu
Output shape: (None, 32, 32, 16)
Conv2D
Activation: linear
Output shape: (None, 32, 32, 16)
BatchNormalization
Output shape: (None, 32, 32, 16)
Activation
Activation: relu
Output shape: (None, 32, 32, 16)
Conv2D
Activation: linear
Output shape: (None, 32, 32, 16)
BatchNormalization
Output shape: (None, 32, 32, 16)

Figure 23: Model w/ modified input layer used in the solution. Note that output has been truncated and continues on the next page.

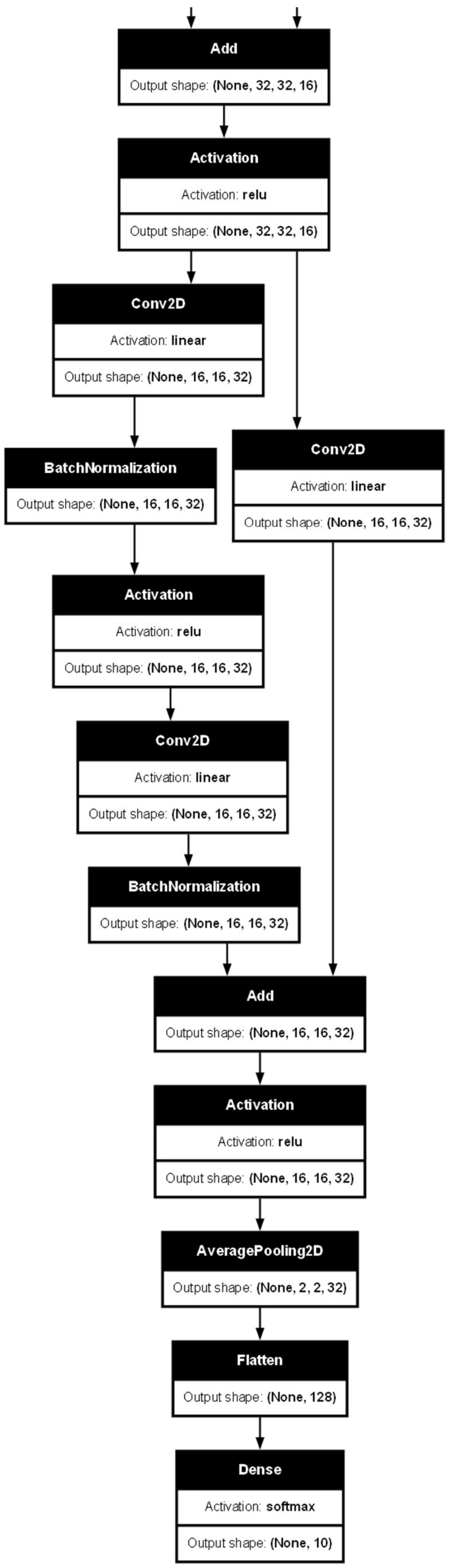

Add
Output shape: (None, 32, 32, 16)
Activation
Activation: relu
Output shape: (None, 32, 32, 16)
Conv2D
Activation: linear
Output shape: (None, 16, 16, 32)
BatchNormalization
Output shape: (None, 16, 16, 32)
Conv2D
Activation: linear
Output shape: (None, 16, 16, 32)
Activation
Activation: relu
Output shape: (None, 16, 16, 32)
Conv2D
Activation: linear
Output shape: (None, 16, 16, 32)
BatchNormalization
Output shape: (None, 16, 16, 32)
Add
Output shape: (None, 16, 16, 32)
Activation
Activation: relu
Output shape: (None, 16, 16, 32)
AveragePooling2D
Output shape: (None, 2, 2, 32)
Flatten
Output shape: (None, 128)
Dense
Activation: softmax
Output shape: (None, 10)

Figure 24: Model w/ modified input layer used in the solution. Diagram continues from previous page.