COMP30027 Report

1. Introduction

With the rapid development of self-driving vehicles, object recognition acts as an important role in those automata to help them obey the traffic rules that are already placed. Therefore, those machine learning models that are embedded in those machines, need to recognise traffic signs accurately and swiftly.

In this project, we have developed 5 machine learning models, 2 Deep Learning, 2 traditional Machine Learning models and 1 model that used the Stacking Ensemble technique. The models include Convolution Neural Network (CNN), Multi-layer Perceptron (MLP), Support Vector Machine (SVM), Multinomial Logistic Regression (MLR), and Stacking model. Then we will evaluate them in terms of performance, analyse models' errors, and see if those models are suitable for autonomous driving systems.

2. Methodologies

2.1 Exploratory Data Analysis

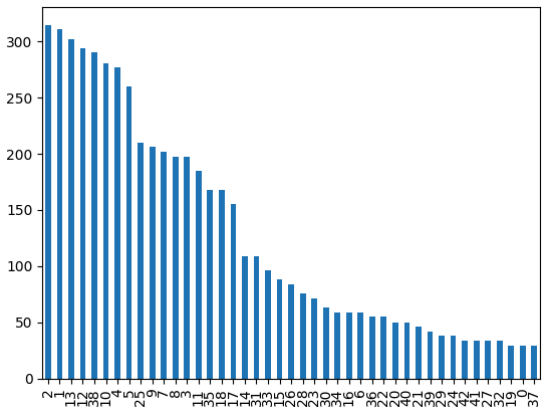

Figure 1: The distribution of classes by the number of instances in the training dataset

Speed Limit of 50 km/h traffic sign (ID of 2) has the most number of instances, which is 315. The go left or right traffic sign (ID of 37) has the least number of instances in the train dataset, with only 29 instances. We can see that the dataset is imbalanced.

The dataset also exhibits various different lighting and capture distance. Notably, there are a few instances in the dataset that have inverse lighting conditions such as image 00028, where the image of the board is not as bright as the background.

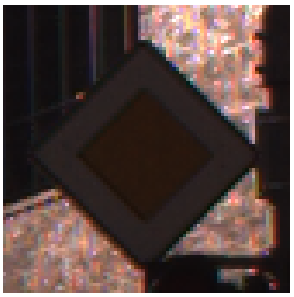

Figure 2: Image 00028.jpg in the training dataset.

The dataset has a wide range of colour intensity, with some images are bright and some are not visible to the human eyes due to poor lighting conditions.

Figure 3: Two images from the training dataset which have a significant difference in visibility.

2.2 Data Processing

We use the following techniques to preprocess images before feeding them to CNN:

- Resize images to (32, 32)
- Data augmentation (rotate, zoom, width shift, height shift)

For traditional ML models, after extracting features from images using a CNN model, we reduce the dimensionality with PCA (to 100 components) and standardize the features. This is essential because SVM, MLR and MLP classifiers are sensitive to data scaling.

2.3 Feature extraction and selection

2.3.1 Handcrafted features

Initially, we tried to extract features directly from the images provided. Given that the dataset comprises traffic signs, we believed that shapes and color attributes would be more discriminative compared to texture features. As a result, we crafted a set of 734 features containing color moments, color histogram, Hu moments, Zernike moments and edge density.

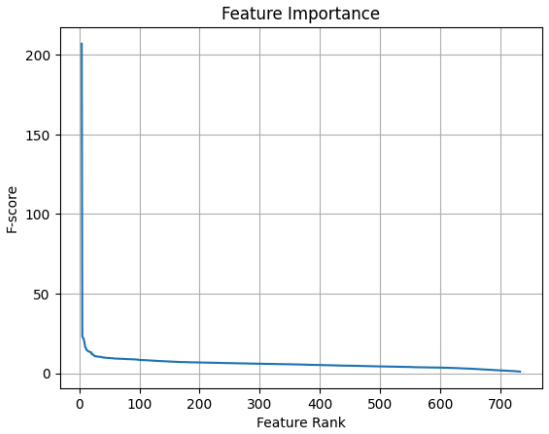

Figure 4: Feature importance ranked by ANOVA F-value.

We then selected the top 50 most important features using ANOVA F-value. After that, Random Forest and SVM classifiers were used. Unfortunately, their performance was unsatisfactory (under 60% accuracy), and therefore the idea was scrapped.

2.3.2 CNN extracted features

Next, we tried using a CNN model to extract more informative and abstract features. Utilizing the same architecture as our CNN model, we extracted 256 features from the penultimate layer. Since CNN is already capable of extracting meaningful features, we only reduce the dimensionality of the feature set with PCA before feeding them to train several classifiers, including MLP, SVM, MLR and Stack Classifier.

2.4 Models

2.4.1 Convolutional Neural Network (CNN)

In this project, we constructed our CNN model from the Sequential method from Keras. Our CNN model architecture consists of 3 main layers:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
batch_normalization (BatchNormalization)	(None, 15, 15, 32)	128
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
batch_normalization_1 (BatchNormalization)	(None, 6, 6, 64)	256
conv2d_2 (Conv2D)	(None, 4, 4, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 2, 2, 128)	0
batch_normalization_2 (BatchNormalization)	(None, 2, 2, 128)	512
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 256)	131,328
batch_normalization_3 (BatchNormalization)	(None, 256)	1,024
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 43)	11,051

Total params: 237,547 (927.92 KB)
Trainable params: 236,587 (924.17 KB)
Non-trainable params: 960 (3.75 KB)

Figure 5: The architecture of the CNN model

2.4.2 Multi-layer Perceptron (MLP)

Our model contains 2 hidden layers with 128 and 64 neurons, one input layer and an output layer using softmax to classify 43 classes. Model regularization was applied through batch normalization and a drop out rate of 0.4. Despite not performing as well as CNN, MLP's performance is superior to traditional ML models.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 256)	25,856
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32,896
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 64)	8,256
dropout_3 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 43)	2,795

Total params: 69,803 (272.67 KB)
Trainable params: 69,803 (272.67 KB)
Non-trainable params: 0 (0.00 B)

Figure 6: MLP architecture

2.4.3 Support Vector Machine (SVM)

SVM acts as a benchmark to compare traditional ML models with the deep learning models that we implemented. While performing poorly when being trained with handcrafted data, the base model's performance significantly improves when being fed CNN extracted data. After conducting grid search to tune the hyperparameters, we found a set of hyperparameters with performance surprisingly close to MLP's : C = 10, gamma = 0.01, kernel = 'rbf'. This indicates that our model uses soft margins and smooth decision boundaries to minimize classification errors.

2.4.4 Multinomial Logistic Regression (MLR)

MLR was evaluated as a traditional baseline classifier, particularly for benchmarking against deeper learning models and models that can capture nonlinear relationships like SVM. Similar to SVM, its performance improves considerably when trained on CNN-extracted features. While its performance was still inferior to that of the MLP and SVM, MLR stood out for its speed and interpretability.

2.4.5 Stacking

The stacking classifier contains 3 base models: MLP, SVM and MLR. The meta-model (Random Forest) is then trained using the predictions from the base models as new features.

3. Result

3.1. Training results

CNN Training result:

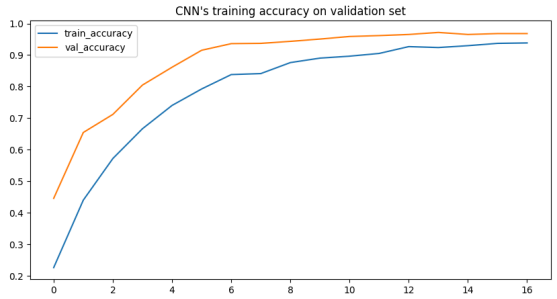

Figure 7: CNN's training and validation accuracy

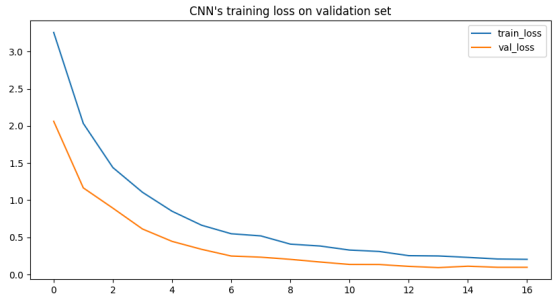

Figure 8: CNN's training and validation loss

Overall, CNN has a good fitting curve as the loss and the accuracy improved significantly as the training epoch progresses. The validation loss curve is lower than the training loss, as well as the validation accuracy is higher than the training accuracy for every epoch; this shows that CNN model has an excellent generalisation.

MLP Training result:

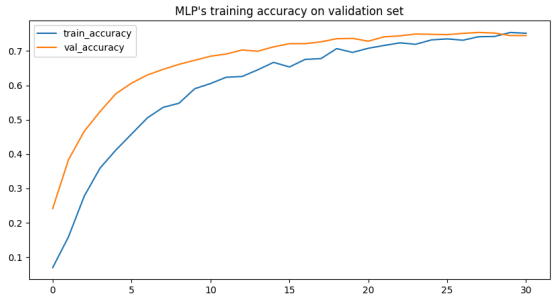

Figure 9: MLP's training and validation accuracy during training epochs.

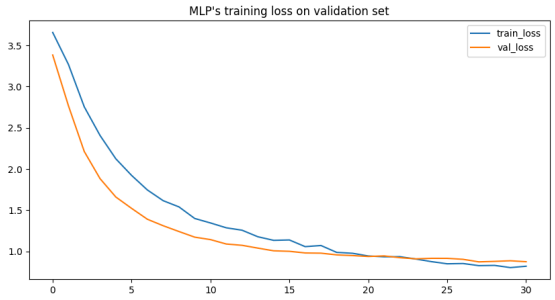

Figure 10: MLP's training and validation loss during training epochs.

MLP training performance exhibits the same training trend at the first 20 epochs. However, both training and validation loss curves meet each other around the 25th epoch, and validation loss is higher than training loss later on. While this is a sign of overfitting, the difference is insignificant, which could be concluded as the MLP has a decent training performance.

3.2. Performance

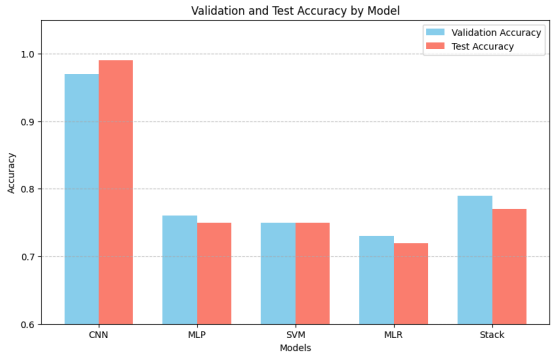

Figure 11: Test and validation accuracy of CNN, MLP, SVM, MLR, and Stacking models.

Model	Accuracy	Macro Average			Weighted Average		
		Precision	Recall	F1-score	Precision	Recall	F1-score
CNN	0.97	0.98	0.97	0.98	0.98	0.97	0.97
Support Vector Machine	0.75	0.82	0.69	0.73	0.77	0.75	0.75
Logistic Regression	0.73	0.73	0.69	0.7	0.73	0.73	0.73
MLP	0.76	0.74	0.68	0.7	0.76	0.76	0.76
Stack	0.79	0.79	0.74	0.76	0.8	0.79	0.79

Figure 12: Machine learning model's classification performance metrics such as accuracy, precision and recall on the validation set.

The model with the best performance is CNN, with 97% accuracy on validation set and 99.5% on test set. MLP achieved 76% on validation set and 75% on test set, while the remaining traditional base ML models' accuracy scores are around 72-73% on validation and test set. After hyperparameter tuning, SVM accuracy improves to 75% on validation and test set whereas MLR does not show any improvement. The ensemble model with MLR, tuned SVM and MLP meta classifiers has 79% accuracy on validation and test set.

CNN has the highest performance with other metrics such as weighted precision and weighted recall, with the score of 0.98 and 0.97 relatively.

On the other hand, multinomial logistic regression has the weakest performance compared to other models, with weighted precision and recall are 0.73.

4. Discussion

4.1 Why do features extracted from CNN yield better results than handcrafted features?

From the results, it is clear that we achieved better results when using the output of the penultimate layer of a trained CNN. This is because CNN is able to automatically learn hierarchical representations from images (Nanni et al., 2017). As CNN progressively learns features using multiple layers, the model captures spatial relationships. As a result,

CNN features are more effective in terms of variations in lighting, perspective distortions, etc. On the other hand, handcrafted features lack adaptability to diverse conditions. If there are variations across pictures like lighting, blurriness, etc, models trained on these features would yield bad results as it fails to generalise well.

4.2 Model Performance Analysis

4.2.1 MLP versus CNN

Even when using the same CNN architecture to extract features from resized 32x32 images, MLP still performs worse than CNN.

An end-to-end CNN model is capable of capturing spatial patterns through convolutional layers, gradually building up complex features by preserving the structure of the image. The end-to-end structure of CNN also allows it to update the weights during training, however, MLP cannot refine this set of features, which might not be optimized. Moreover, while MLP receives features extracted from CNN, they are flattened into a one-dimensional vector, which removes the explicit spatial layout. As a result, MLP processes the input as an unstructured set of values, without access to the spatial relationships that the CNN originally learned. What's more, flattening CNN output into 256 dimensional vector and then applying PCA, standardization may have over compressed information, leading to the loss of subtle patterns. This limits the MLP's ability to learn the image's structural information, leading to lower classification performance.

However, using CNN-extracted features still significantly improves MLP performance compared to feeding raw images directly into an MLP. Due to the inductive bias of MLP, it cannot learn the images' spatial information. Therefore, turning images into a 1D array and feeding directly into MLP would cause another issue: translation invariant (i.e the model cannot classify images from the same class but with different lighting, position, etc). MLP is also highly computationally inefficient in this case as each pixel requires its own input neutron, which greatly increases the number of parameters and might lead to overfitting.

4.2.2 SVM versus MLR

SVM outperforms MLR when training on CNN-extracted features, with accuracy rates of 76% versus 69% respectively. While SVM can model non-linear decision boundaries, MLR only assumes that the classes are linearly separable in feature space. Using t-distributed stochastic neighbor embedding (t-SNE) to visualize the features on a 3D space, we can see that the features are possibly non-linear as the classes form complex shapes and have substantial overlapping.

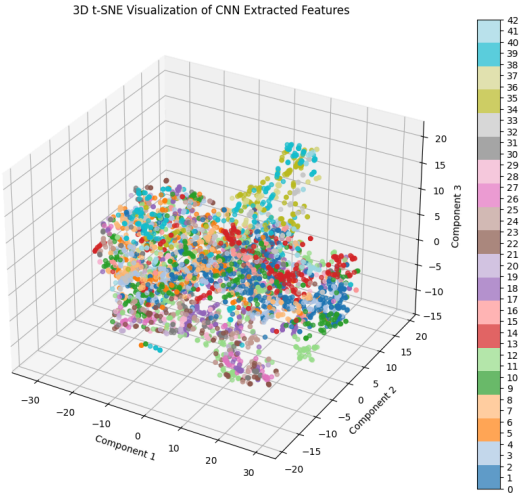

Figure 13: 3D t-SNE visualisation of CNN extracted features.

This means that the CNN-extracted features retain significant nonlinear structures that require a model like SVM to capture class boundaries effectively. Furthermore, the superior performance of SVM with RBF kernel compared to SVM with linear kernel provides strong evidence that these features exist in a non-linearly separable space.

4.2.3 Deep Learning Models versus Traditional ML models

Traditional ML models perform significantly worse than that of models like CNN. This is because the features extracted from CNN are too complex and entangled, therefore, models that can handle complex interactions among features like CNN tend to perform better.

Interestingly, MLP's performance is on par

with that of SVM. This might be because we have an imbalanced dataset, which MLP especially struggles with. MLP tends to overfit majority classes and underperforms on minority classes. This can be observed using the confusion matrix of the model.

4.2.4 Stacking model analysis

Our stacking model was able to perform better than any of the base models alone as it was able to take advantage of the strength of each model:

- SVM is strong at finding non-linear decision boundaries.
- MLP is able to capture complex non-linearities missed by SVM using hidden layers.
- MLR, despite being a linear model, performs reliably in regions where decision boundaries are approximately linear.

The performance improvement demonstrates that the model captures a more complete representation of the decision space than any single algorithm, which effectively reduces both bias (from the linear model) and variance (from the more complex models) compared to any individual approach. This diversity allows the stacked model to handle the structure of the CNN-extracted features and make more accurate predictions.

4.3 Error analysis

CNN		SVM		MLP		Logistic Regression		Stack model	
True class	Predicted class	True class	Predicted class	True class	Predicted class	True class	Predicted class	True class	Predicted class
5	8	8	5	5	3	1	2	8	5
42	41	7	8	8	5	5	3	5	3
41	9	4	8	2	5	8	5	2	1
38	8	3	5	2	1	2	5	2	5
36	9	2	5	10	5	5	8	1	2
26	13	10	5	1	2	3	5	10	5
20	23	2	8	5	10	1	4	4	1
19	26	36	38	4	1	8	7	2	8
10	9	40	38	3	5	8	4	31	18
9	32	20	25	2	7	5	10	12	6

Figure 14: Top 10 misclassified classes by different machine learning models

Signs with similar colors and overall shape, but with different numbers/symbols were the most commonly misclassified among the top confused classes. An example of this confusion is class 5 being consistently misclassified as class 8 across multiple models, as both belong to the speed limit category but indicate different speed values.

Figure 15: One instance of class 5 and class 8 which has been misclassified by CNN

Since they belong to the same semantic category, they are represented closely in the model's feature space. Additionally, in photos with lower resolutions and contrast, or are distorted with blur, the models struggle to distinguish them. In such cases, models tend to select classes that are better represented (all traditional models and MLP misclassify class 8 with class 5).

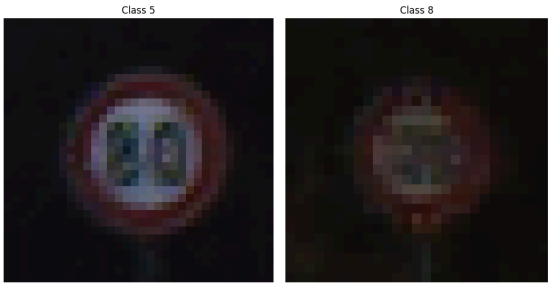

Figure 16: One instance of class 5 and class 8 which has been misclassified by MLP

It is also noteworthy that the misclassification patterns of the stack model closely resembles that of other ML models and MLP, which is expected given that they are used as the base models.

It is also apparent that the models sometimes mistake one class with another due to similarities in numbers or symbols. For example, in MLP, class 2 (Speed Limit of 50 km/h) images are usually mistaken with class 5 (Speed limit of 80 km/h). This could be due to the fact that the number "8" and the number "5" have almost the same drawing stroke, which both have a right curve on the right bottom side, and a left curve on the top left side.

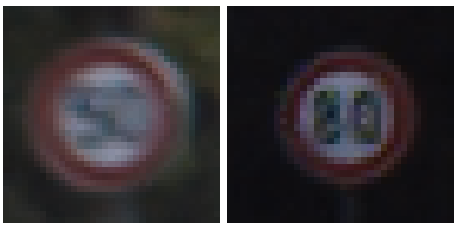

Figure 17: One instance of class 2 and class 5 which has been misclassified by MLP

Confusion Matrix

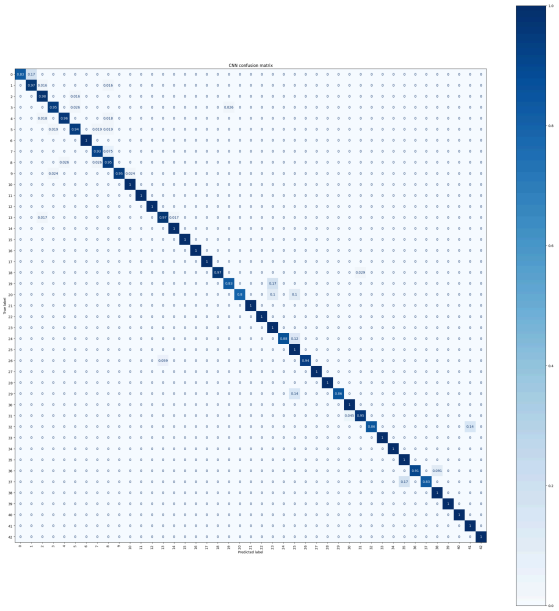

Figure 18: CNN's row-normalised confusion matrix (normalisation by true value)

As expected, CNN model is able to classify with high accuracy with all of the classes. However, the model seems to underperform when predicting instances from class 20 or 19. It could be traced back to the fact that those classes have the least amount of instances in the training dataset.

Another possible reason is that instances are low-quality and have many noises.

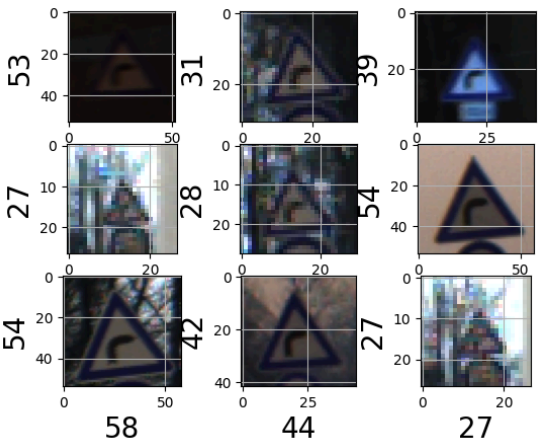

Figure 19: 9 randomly-picked instances of class 19

For example, these random instances of class 19 that are picked from training dataset, the pictures of this traffic sign are low-resolution. These specific signs usually go with other traffic signs, which poses a minor obstacle for CNN to effectively recognise.

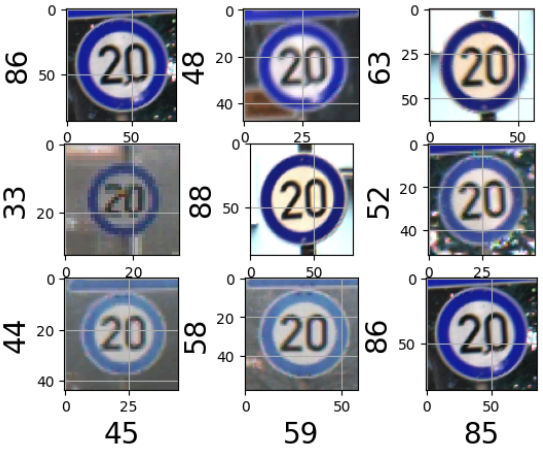

Figure 20: 9 randomly-picked instances of class 0

Comparatively, while class 0 also has a similar number of instances in the training dataset, the images are usually brighter, with higher resolution. The images are not as quite blurry with as much noise.

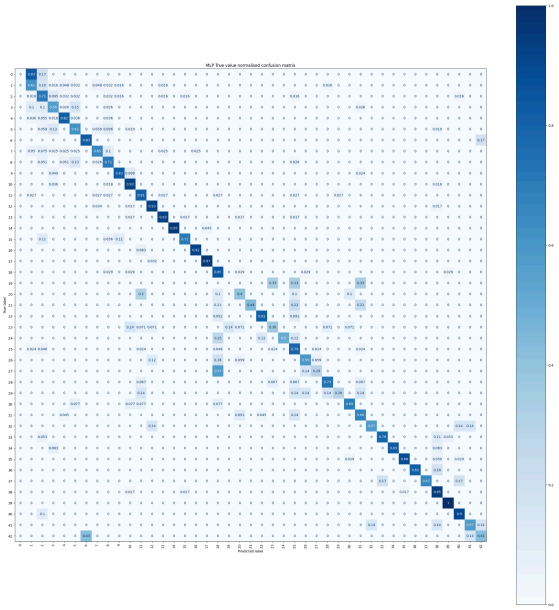

Figure 21: MLP's row-normalised confusion matrix (normalisation by true value)

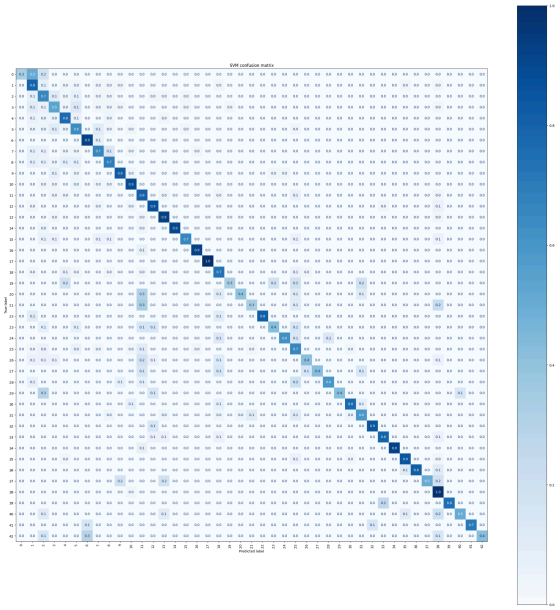

Figure 22: SVM's row-normalised confusion matrix (normalisation by true value)

The MLP model was not able to correctly classify instances of class 0 and class 19. We believe that this is because of the lack of training data for these classes. Due to this underrepresentation, MLP may have ignored them in favor of minimizing the loss across more frequent classes (Most of class 0 instances are misclassified as class 1- a highly represented class).

Interestingly, this problem does not exist in SVM and MLR despite having low prediction accuracy in these classes. As a result, the stack model was able to leverage this and improve its predictions.

The diversity in model predictions evidently leads to more accurate predictions of each class. For lower-represented classes, this is especially beneficial, as some models may still capture useful patterns that others miss. For instance, while the MLR model performs best on class 19, SVM achieves a recall of 0.3 and the MLP fails to correctly classify it (recall = 0). By leveraging the strengths of each model, the stacked model significantly improves performance on class 19, increasing recall to 0.5.

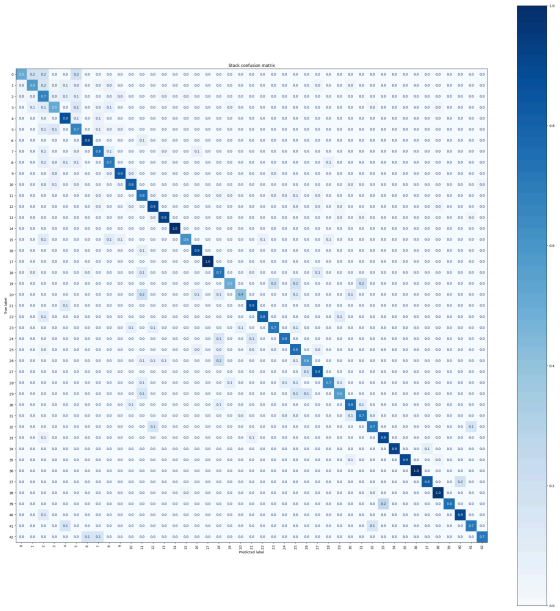

Figure 18: Stack's row-normalised confusion matrix (normalisation by true value)

5. Conclusions

In this project, we have explored and compared various machine learning models and their performance in terms of classifying traffic signs from the GTSRB dataset. The experiment reveals that the convolutional neural

network has a significant performance over traditional machine learning models in accuracy, precision, and recall metrics. The CNN's outstanding performance can be attributed to its ability to

While models such as MLP and SVM seem to have improved on their performance, they still feel short compared to CNN due to its inability to recognise spatial features due to flattening feature vectors.

The stacking ensemble model, combining MLP, SVM, and MLR, with Random Forest as the final estimator, proved to improve the overall accuracy, precision, and recall metrics by leveraging the strengths of individual models - despite not surpassing CNN in terms of efficiency.

Our error analysis reveals that the top misclassified class across five models often have similar traits such as colour, font strokes, and shapes. The similarities pose a significant obstacle for these models to classify correctly.

6. References

Nanni, L., Ghidoni, S., & Brahnam, S. (2017). Handcrafted vs. non-handcrafted features for computer vision classification. *Pattern Recognition*, *71*, 158–172. https://doi.org/10.1016/j.patcog.2017.05.025

Saleh, R., & Fleyeh, H. (2022). Using Supervised Machine Learning to Predict the Status of Road Signs. *Transportation Research Procedia*, *62*, 221–228. https://doi.org/10.1016/j.trpro.2022.02.028