GTSRB Classification
COMP30027 Report
(~1700 words minus titles, data and references)

1. Introduction

In 2025, convolutional neural networks (CNNs) are said to "lead the charge" (Alvi, 2024) when it comes to image classification tasks. For the GTSRB traffic sign classification task, we hence sought to utilize a CNN as well as other ML models such as a Random Forest (RF) and Support Vector Machine (SVM) and compare their effectiveness for labelling images. Accuracy is of utmost importance for real world applications of our classification task (e.g. self-driving cars), so we stacked each model we built and chose optimal parameters to make our final predictor as reliable as possible.

2.0 Methodology

Tackling this task effectively involved creating a robust workflow where data processing errors did not occur and hard coding is discouraged. To do this we broke the task up into these subtasks:

2.1 Feature Extraction

To distinguish between classes, we engineered additional features from the raw image data to capture more discriminative information for classifiers to utilise.

Firstly, we extracted features to represent shape-based characteristics of the images. To do this we pre-processed the images to ensure our feature extraction methods could distinguish these shape-based features as best as possible. This included applying filters, blurs, masks as well as enhancing contrast. Preprocessing was slightly different for each feature engineered to ensure the best results; this was done iteratively by visually evaluating effects on feature distinctiveness.

For shape features we utilised OpenCV to extract:

1. Contours (curves or boundaries for the subject of the image) and passed their associated properties such as aspect ratio and solidity into our model.
2. Hu Moments (seven shape descriptors which are transformation invariant), these are calculated from the statistical properties of pixel intensities, helping identify shapes even if the subject is rotated, scaled or translated.
3. Edge Orientation Histograms: This feature captures the distribution of edge directions of the subject in our image. It utilises edge detection algorithms which calculate gradient direction at edge pixels in the image and group them into bins. Given that traffic signs are characterised by their well-defined boundaries that can possibly not change based on class, this feature provided beneficial information to help models distinguish signs.

Lastly, we extracted features from the HSV colour space to provide our models with more robust colour information. HSV separates colour from brightness to give our models information about the colour of signs in low light photos.

2.3 Feature Selection

Before training, we selected the most informative features using mutual information as we had an abundance of continuous features, some of which were not strongly informative (noisy). We also removed highly correlated features as information extracted from them can be extracted from other features instead.

Feature:	MI
hog_pca_0	0.882652
hog_pca_3	0.796677
Edge_Hist_Bin_6	0.596259
Edge_Hist_Bin_2	0.566049
hog_pca_1	0.554871
Edge_Hist_Bin_7	0.530975
Edge_Hist_Bin_3	0.519416
hog_pca_2	0.493234
Edge_Hist_Bin_5	0.429996
H_hist_bin_16	0.40201

Figure 2.3.1- Most informative features, sorted by mutual information.

2.4 Model Selection / Validation

We then chose to create 3 base models to train and stack using a meta classifier. We trained each model using the same (K=5) fold validation splits to ensure no data leakage.

1. SVM – Good for separating classes in high dimensional feature spaces like ours with the features we created.
2. RF – Robust to noise, overfitting, no extra preprocessing/scaling needed. Easy and fast to train and perform well on a large range of feature types which our dataset has.
3. CNN – Models complex spatial patterns in the image data, layers extract features that are more informative than those we created for the other models.

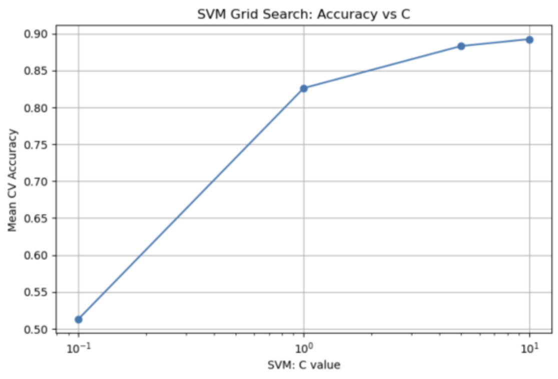

Figure 2.4.1- SVM Accuracy VS Hyperparameter: C

As seen in figure 2.4.1, we used grid search to tune each model (like SVM) to predict accurately and not overfit. In this case we chose C = 5 to balance accuracy and efficiency and not risk overfitting.

After training, we extracted the prediction probabilities from each validation fold for each of the models, these validation sets hadn't been seen by each model and hence data leakage was avoided. These were then fed into a Logistic Regression metamodel, which learnt how to optimally combine our base model predictions by identifying patterns in the way each model classified instances. We chose this approach to ensure a reduction in bias and variance and **better generalisation** from our original base models.

Lastly, to assess model performance we calculated Bias, Variance, Categorical Cross-Entropy Loss and Accuracy and tweaked hyperparameters for best performance. For our SVM which does not directly output prediction probabilities, we calculated cross-entropy loss using probabilities given by sklearn, which feeds the distances from the hyperplane into a logistic regression model to find probabilities for predictions.

3. Results

3.1 SVM Performance

Fold	Accuracy	Bias	Variance	Cross-Entropy Loss
1	0.8634	0.071	0.0142	0.5618
2	0.8807	0.0694	0.01387	0.5519
3	0.8789	0.0681	0.01393	0.5185
4	0.8651	0.0702	0.01411	0.5385
5	0.8678	0.0701	0.01391	0.5491
Mean	**0.8712**	**0.0698**	**0.014**	**0.544**

Figure 3.1.1- SVM Model Accuracy, Loss, Bias & Variance across Validation folds.

3.2 RF Performance

Fold	Accuracy	Bias	Variance	Cross-Entropy Loss
Fold 1	0.8306	0.1012	0.006	1.0685
Fold 2	0.8388	0.1018	0.0058	1.0852
Fold 3	0.8324	0.1017	0.0059	1.0725
Fold 4	0.8295	0.1025	0.0057	1.0915
Fold 5	0.8177	0.1037	0.0057	1.1137
Mean	**0.8298**	**0.1022**	**0.0058**	**1.0863**

Figure 3.2.1- RF Model Accuracy, Loss, Bias & Variance across validation folds.

3.3 CNN Performance

Fold	Accuracy	Bias	Variance	Cross-Entropy Loss
Fold 1	0.9617	0.037	0.0217	0.188
Fold 2	0.9545	0.039	0.0216	0.16
Fold 3	0.9608	0.0371	0.0216	0.1303
Fold 4	0.9572	0.0399	0.0215	0.1875
Fold 5	0.9663	0.0345	0.0217	0.1341
Mean	**0.9601**	**0.0375**	**0.0216**	**0.16**

Figure 3.3.1- CNN Model Accuracy, Loss, Bias & Variance across Validation folds.

3.4 Stacking Model Performance

Fold	Accuracy	Bias	Variance	Cross-Entropy Loss
Fold 1	0.9763	0.0293	0.0216	0.0973
Fold 2	0.9709	0.0314	0.0217	0.1105
Fold 3	0.9754	0.0298	0.0215	0.0959
Fold 4	0.9708	0.0309	0.0215	0.1043
Fold 5	0.9754	0.0306	0.0216	0.1054
Mean	**0.9738**	**0.0304**	**0.0216**	**0.1027**

Figure 3.4.1- Stacking Meta-model Accuracy, Loss, Bias & Variance across Validation folds.

Test Data Accuracy: 0.973

4. Discussion, Model Comparison & Critical Analysis

4.1 SVM

Our SVM model was the most accurate (0.87 mean) base model that we trained on our extracted features due to its strengths in handling high dimensional numeric feature sets. Since our strongest features were binned and high dimensional such as edge histograms and HOG descriptors, an SVM model was well suited for their classification due to their ability to separate high dimensional data. Our relatively high accuracy value as well as low variance (0.014) supported this.

By employing a Radial Basis Function (RBF) kernel, non-linear patterns in our data were best identified and visually similar traffic sign images were made linearly separable. Through grid search, C=5 (regularisation style parameter) emerged as the optimal for balancing complexity and overfitting. A smaller C risked too general decision boundaries and underfitting (high bias), while anything larger only improved performance minorly and risked overfitting to noise (high variance). Hence our selected value helped find a good trade-off to minimise both bias (0.0698) and variance (0.014). Contributing to strong generalisation which is reflected by our stability in accuracy across validation folds, and relatively low cross-entropy loss of (0.544) as opposed to our RF. The low entropy loss indicates that the SVM made confident predictions, hence it proved itself as a reliable base model with consistent outputs.

4.2 RF

The Random Forest classifier achieved a lower mean accuracy (0.83) than SVM; however, it demonstrated its strengths in making stable predictions. The sklearn library RF's ensemble of decision trees utilise bagging/random sampling to reduce overfitting, this is evident with our low variance of 0.0058. This indicates that the model's predictions were stable over each validation fold and random sampling successfully reduced correlation between base trees, even when faced with overlapping patterns or noise within classes.

Apart from higher stability, the RF classifier was still our worst model in terms of performance, with strong bias (0.102) and higher cross entropy loss (1.09). Indicating that the model was less confident in its predictions and often misclassified instances. Suggesting that even though the RF effectively reduced variance, it struggled to capture complex patterns in our data.

Despite efforts to try and tune the model by adding more trees and maximising depth, it was unable to achieve the accuracy of our other models, likely due to the RF' s splitting criteria and aggregation method being less compatible with our highly dimensional dataset. RF's reliance on weak learners likely led to oversimplified decision boundaries and hence worse accuracy & bias.

4.3 CNN

The CNN was by far the best classifier out of our base models, as expected due to its ability to learn hierarchical features from raw image data. To leverage this, we set up the network with 3 convolutional layers, separated by pooling layers, this ensured that extracted features were amplified and computation was cut down. The CNN was therefore able to extract and predict off not only specific aspects like the edges and gradients which we extracted for the other models, but also more complex, hierarchical patterns like spatial relationships and textures. Therefore, it captured richer information, enabling it to achieve significantly higher accuracy (0.96). In addition to this, the CNN also achieved the lowest cross-entropy loss (0.16), indicating that it was the most confident base model with the most trustworthy predictions. This is impressive as it was done with limited hyperparameter tweaking as well, we were unable to utilise grid search due to time constraints and had to only adjust the number of kernels in each layer as well as kernel size. Given more time, experimenting with more values and adjusting other parameters such as the dropout rate to balance accuracy with overfitting would be ideal. Although overfitting was well controlled in the final model with variance being the lowest of all base models, indicating this model was likely the best at dealing with noise.

4.4 Stacked Model

Lastly, as expected the stacked model leveraged the strengths of each base model and performed best on separate random unseen validation folds in all metrics. As the metamodel was able to learn weights for each base model's predictions, it was able to correct systematic errors made by base models, hence leading to minimised bias (0.03) and variance (0.02), and a mean accuracy of 0.974 on validation folds and 0.973 on test data. Since both accuracies are similar, we can infer that the model is therefore robust and good at generalising to unseen instances. This importantly indicates to us that the model has not overfitted to the training data, something we actively sought to avoid by using a regularisation strength of C=10, in our logistic regression meta model. Since it is quite a common occurrence for highly dimensional noisy data to be overfit. Ultimately indicating that our stacking classifier adequately exploited the base CNN's raw accuracy and combined this with the low variance and stability of our other base classifiers.

5. Conclusions

Overall, we developed a comprehensive approach to the GTSRB classification problem that effectively generalised to unseen data. Our approach involved engineering features that extracted information out of images and fed less noisy and more concise data to our models, as hyperparameter tuning to ensure overfitting and underfitting was minimised. Using a logistic regression meta model, we exploited the strengths of each base classifier such as a CNN's ability to extract complex features and the SVM's ability to separate high dimensional data to create a robust and reliable stacking classifier, with low bias and variance and the higher accuracy than our base models. In future attempts, more extensive feature engineering and hyperparameter tuning could help us improve results further.

6. References

- Alvi, F. (2024, March 20). *Image classification in 2025: Insights and advances*. OpenCV. https://opencv.org/blog/image-classification/