A Study of Traffic Sign Recognition with Classical and Deep Models: Performance and Error Analysis

Yi Ding

1. Introduction

The goal of this project is to build supervised Machine Learning methods for classifying traffic signs with a subset of German Traffic Sign Recognition Benchmark (Stallkamp et al., 2011). Traffic sign classification is crucial in driver-assistance systems and autonomous vehicles, where reliable performance under real-world conditions is essential.

This project addresses the challenge of classifying sign types from natural images, which may contain noise, distortions, or complex backgrounds. The primary objective is to design and evaluate pipelines for both classic models that require manual feature extraction and selection, and neural networks that have automated feature extraction.

To accomplish this objective, we engineered features including RGB colour histograms, HSV (hue, saturation, and value), Histogram of Orientated Gradients (HOG), Local Binary Patterns (LBP), and Canny contours. After feature selection, we developed and evaluated three classifiers: KNN, Random Forest, and a CNN based on LeNet-5.

2. Methodology

2.1 Dataset Observations

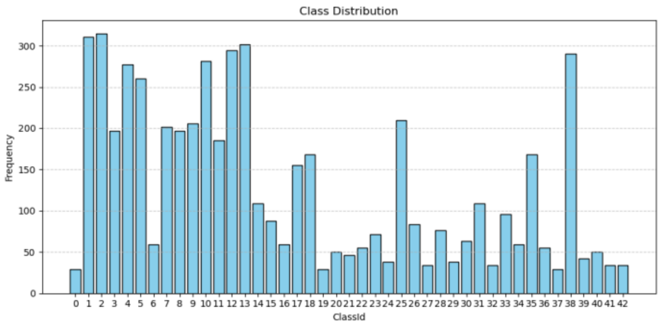


Figure 1– Histogram of class distribution for the training set.

For the training set, there are 5488 instances and 43 classes with highly uneven class distribution (Figure 1). This may cause trained classifiers to become biased toward majority classes and perform poorly on minority classes.

Threshold	32x32	64x64	128x128	256*256
Frequency	687	4373	5397	5488

Table 1- Cumulative frequency for images below each image size threshold.

Most images in the training set are small, with dimensions smaller than 64x64 pixels (Table 1), and therefore lack fine-grained details. Moreover, the images vary in size, which can cause inconsistencies in feature dimensionality for methods like HOG.

2.2 Feature Engineering

2.2.1 Image Preprocessing

Before feature extraction, resizing and noise reduction were applied. The purpose of resizing was to address the issue identified in section 2.1, ensuring that the extracted features had consistent dimensionality.

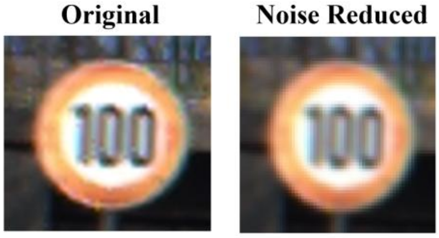


Figure 2- Comparison between original and noise reduced image using Gaussian blur. Image ID: 6867.

Upon inspection, many images in GTSRB were not cropped to exclude the background. To reduce the impact of background noise during feature extraction, Gaussian blur was applied (Figure 2).

2.2.2 Feature Extraction

The provided feature set was not used because the provided HOG features had already been PCA-reduced, which could cause data leakage if used for validation.

We engineered RGB and HSV histograms to capture traffic sign colours, HOG for structural information and symbol recognition, LBP for texture patterns, and contours for shape-based features such as the number of vertices and circularity ratio. Contours were extracted using the Canny edge detection algorithm (Canny, 1986).

To reduce noise caused by colour variance, images were converted to grayscale for HOG, LBP, and contour extraction.

2.3 Preprocessing and Learners

Before hyperparameter tuning, an 80%-20% train-test split was applied to the training set. To reduce model and evaluation bias, stratified sampling was used because it preserves the original class distribution.

2.3.1 KNN Pipeline (Benchmark)

KNN does not make assumptions about the data distribution; therefore, it was expected to perform well for multiclass image classification, where the decision boundary is likely complex.

Due to the "curse of dimensionality," KNN's performance was expected to decline with the current feature space dimensionality of 1966. Therefore, a filtering method was used to select the top 50 features with the highest mutual information scores. Then, normalization was applied to ensure features were on the same scale and contributed evenly to the distance calculations. To further reduce dimensionality and improve model generalization, PCA was applied to the scaled features with a threshold of 0.9 for total variance explained.

2.3.2 Random Forest Pipeline

It was observed that the KNN model had low accuracy due to information loss caused by dimensionality reduction during feature selection (explained in Section 4.2). Therefore, Random Forest was used because it is more robust to high dimensionality.

To address the class imbalance issue identified in Section 2.1, balanced class weights were employed so that misclassifications of minority classes would be penalized more heavily during tree construction, preventing their underrepresentation.

To reduce noise from irrelevant features, mutual information scores were calculated to select the top 300 features (see Section 5.1 for limitations).

2.3.3 CNN Pipeline

CNN was used due to its advantages in automated image feature extraction and end-to-end learning.

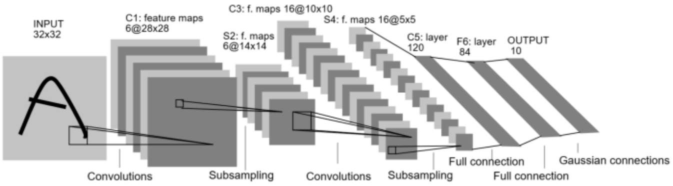


Figure 3– Original architecture for LeNet-5 (from Lecun et al., 1998)

The LeNet-5 architecture proposed by LeCun et al. (Figure 3) was adopted because it was designed for the classification of small 32×32 images, which aligns well with the GTSRB dataset (Table 1).

For CNN image preprocessing, normalization was applied using the mean and standard deviation of RGB values from the training set. By rescaling all images to a comparable scale, the CNN trained more stably, with minimal fluctuations in validation accuracy across epochs (Figure 5).

For regularisation, early stopping was used with a patience level of 10, based on validation accuracy. This helped reduce the risk of overfitting by halting training once the validation accuracy stabilized.

2.4 Hyperparameter Tuning

Model	Tuned Hyperparameters
KNN	Number of neighbours: 3, 5, 7, 9 Voting strategy: majority, inverse distance
Random Forest	Number of trees: 50, 100, 200 Feature sub-sample size $c * (\log_2 F + 1)$ where F is the number of selected features: $c \in \{0.5,1.0,1.5,2.0\}$
CNN	Batch size: 32, 64, 128 Learning rate: 0.01, 0.001, 0.0001 Number of epochs: [0, 200]

Table 2– Sets of hyperparameters for tuning.

Grid search was applied to find the optimal set of hyperparameters (Table 2). To reduce evaluation variance, 5-fold cross-validation was used for KNN and Random Forest. A single holdout set was used for CNN, as CNN models have a high computational cost during training.

3 Results

3.1 KNN Validation

		Number of neighbours			
		3	5	7	9
Voting strategy	Majority	0.7196	0.7312	0.7248	0.7271
	Inv Distance	0.7405	0.7469	0.7412	0.7419

Table 3– KNN cross-validation accuracy scores.

The number of neighbours was chosen to be 5 with inverse distance voting strategy (Table 3).

3.2 Random Forest Validation

		Number of trees		
		50	100	200
Max features	0.5	0.9050	0.9212	0.9271
	1.0	0.9121	0.9228	0.9328
	1.5	0.9153	0.9264	0.9310
	2.0	0.9175	0.9216	0.9244

Table 4– Random Forest cross-validation accuracy scores.

Number of trees and max features were chosen to be 200 and $1.0 \cdot (\log_2 300 + 1) \approx 9$ respectively (Table 4).

3.3 CNN Validation

		Batch size		
		32	64	128
Learning rate	0.01	0.8306	0.9044	0.9508
	0.001	0.9572	0.9581	0.9545
	0.0001	0.9536	0.9362	0.9490

Table 5– CNN validation accuracy scores.

Batch size and learning rate were chosen to be 64 and 0.001 respectively (Table 5).

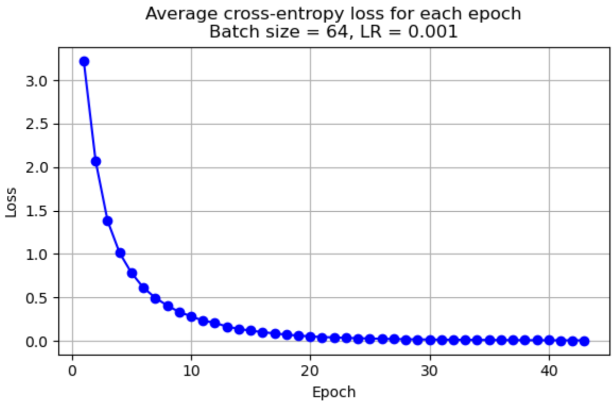


Figure 4– Average cross-entropy loss for each epoch of CNN training.

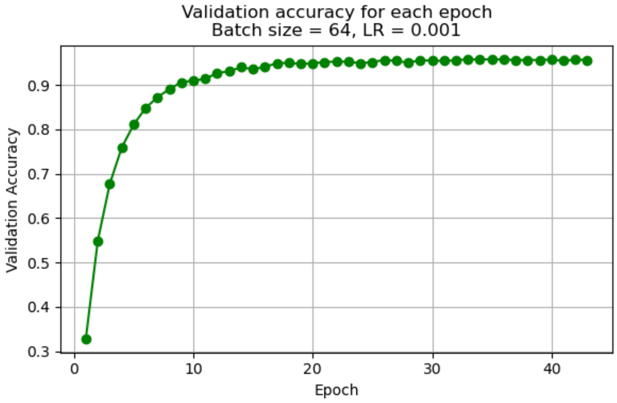


Figure 5– Validation accuracy score for each epoch of CNN training.

Early stopping was triggered at 43rd epoch. It appeared the model's average cross-entropy loss and validation accuracy both stabilized around 35th epoch (Figure 4, 5).

3.4 Test Results

KNN	Test Set	Kaggle Test
Accuracy	0.76	0.75
Precision (Weighted Average)	0.76	N/A
Recall (Weighted Average)	0.76	N/A
F1-score (Weighted Average)	0.76	N/A
Random Forest	Test Set	Kaggle Test
Accuracy	0.94	0.94
Precision (Weighted Average)	0.95	N/A
Recall (Weighted Average)	0.94	N/A
F1-score (Weighted Average)	0.94	N/A
CNN	Test Set	Kaggle Test
Accuracy	0.97	0.96
Precision (Weighted Average)	0.97	N/A
Recall (Weighted Average)	0.97	N/A
F1-score (Weighted Average)	0.97	N/A

Table 6– Test results for KNN, Random Forest, and CNN.

Both Random Forest and CNN exceeded the benchmark set by KNN for accuracy, weighted precision and recall. CNN had slightly better performance than Random Forest (Table 6).

4 Discussion and Critical Analysis

Since the distribution of classes is highly unbalanced (Table 1), each model's performance on certain classes was evaluated based on their precision and recall rather than the number of instances misclassified.

4.1 Results Discussion

All three models appear to generalize well, as they achieved similar cross-validation accuracy, test accuracy, and Kaggle test accuracy, with balanced weighted precision and recall (Table 6).

Fundamentally, KNN was more reliant on manual feature extraction, as it does not learn or extract key information during training. However, real-world traffic sign images tend to contain complex features with high dimensionality (Section 2.2.2). Therefore, the KNN model's lower accuracy (Table 6) may be explained by the loss of variance during feature selection.

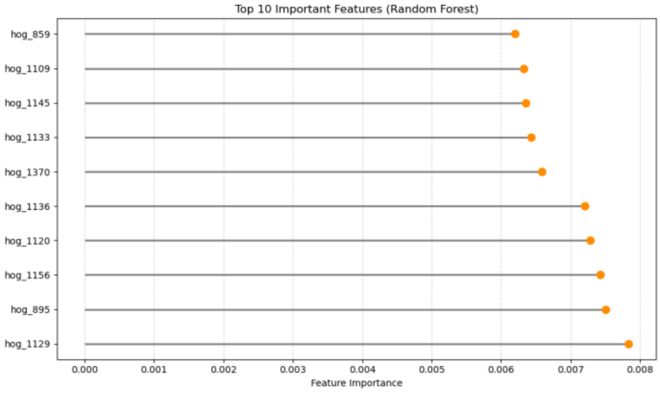


Figure 6– Lollipop chart for feature importance calculated based on features' contributions to node splitting for Random Forest.

In comparison, Random Forest is more robust to high dimensionality, as it considers only a subset of image features to split on at each node. Since Random Forest selects features that reduce node impurity, this allows individual trees to favour more informative image features during training (Figure 6), unlike KNN, where the selected and scaled features are treated equally in distance calculations.

Since CNN uses convolutional layers to learn local patterns, it has the advantages of automated traffic sign feature extraction and end-to-end learning, whereas manual feature extraction for Random Forest may be limited by human intuition. However, CNN has lower interpretability compared to Random Forest due to its complex hierarchical layered structure.

4.2 KNN Error Analysis

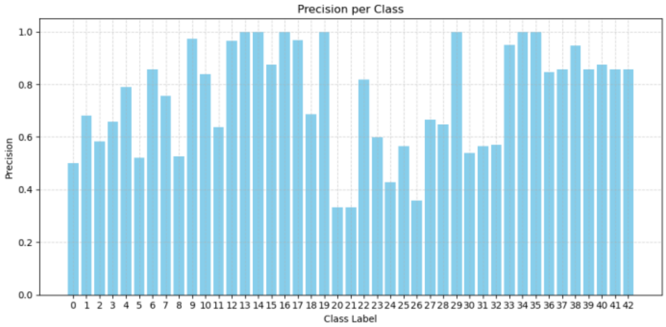


Figure 7– Histogram of KNN precision for each class based on test set.

We will use a threshold of 0.4 to identify classes with poor precision; here, KNN model had particularly low precision for classes 20, 21, and 26 (Figure 7).

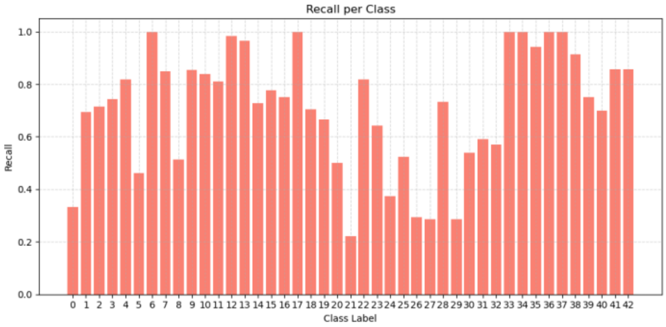


Figure 8– Histogram of KNN recall for each class based on test set.

Furthermore, KNN had particularly low recall for classes 0, 21, 24, 26, 27, 29 (Figure 8).

We hypothesize that the low precision and recall were primarily caused by information loss during the feature selection process.

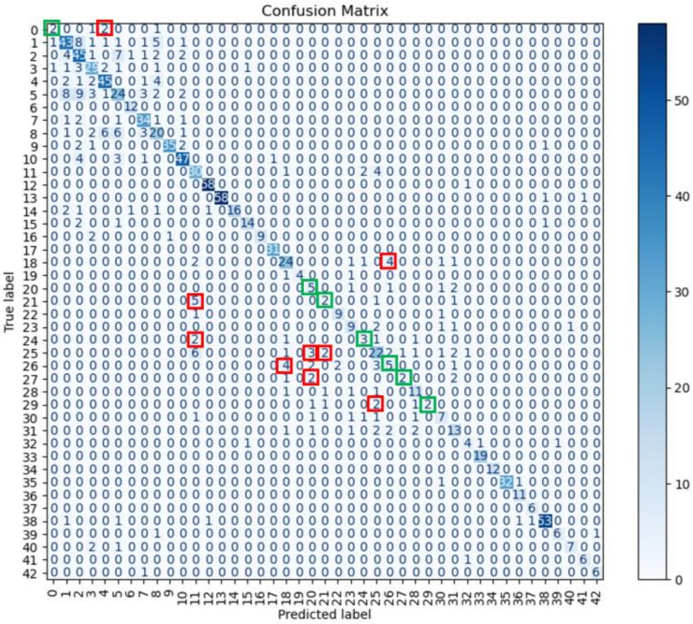
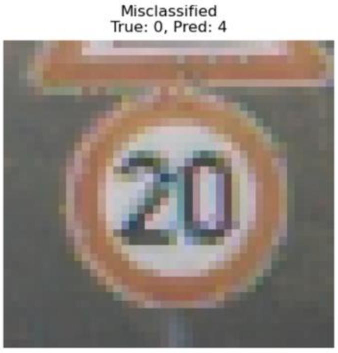


Figure 9– Confusion Matrix for KNN test results, with major misclassifications for low precision and low recall classes highlighted in red.

There appeared to be major confusions (Figure 9) between classes 0 and 4, 21 and 11, 24 and 11…



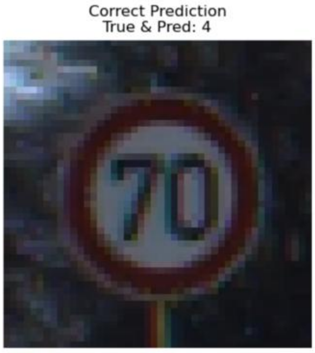


Figure 10– Example of misclassification for class 0. Left image ID: 7393. Right image ID: 228.

After inspecting examples of misclassifications, it appeared that the misclassified signs all had similar shapes and colour schemes compared to the predicted classes. For example, signs in classes 0 and 4 both share a red, white, and black colour scheme with a circular shape (Figure 10). The primary pattern that distinguishes these signs is the symbol or number. This suggests that colour-based, texture-based, and shape-based features were not sufficiently discriminative in these cases.

We defined confidence values for KNN as the predicted class's contribution to the total weighted distance from all neighbours.

	Misclassified	Correct Prediction
Confidence	0.5989	1.0000
Percentile	27.5%	78.23%

Table 7– Confidence value and the percentile of the confidence among all predictions for Figure 10 examples.

Low confidence values were observed for misclassified examples (Table 7), which suggests that the neighbourhoods for low-precision and low-recall classes were ambiguous in the current feature space. This indicates that the selected features were likely not expressive enough to distinguish these classes, which aligns with our earlier hypothesis. This is a limitation of KNN, as it requires a low number of features to avoid the "curse of dimensionality."

A potential improvement would be to use embeddings from the CNN model to provide more expressive features for the KNN model.

4.3 Random Forest & CNN Error Analysis

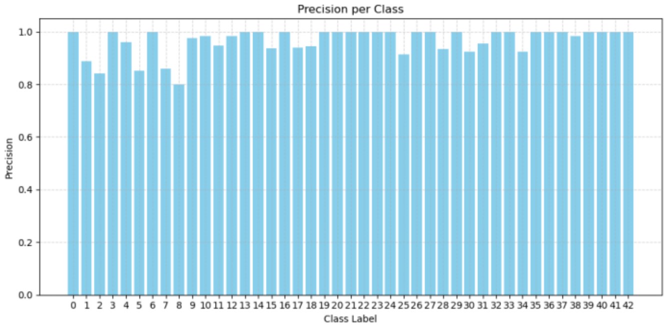


Figure 11– Histogram of Random Forest precision for each class based on test set.

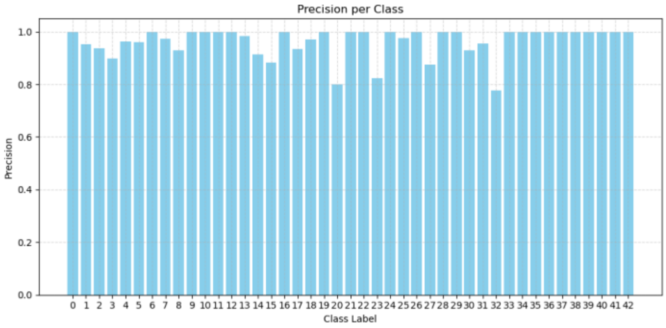


Figure 12– Histogram of CNN precision for each class based on test set.

Random Forest and CNN had high precision for all classes (Figure 11, 12).

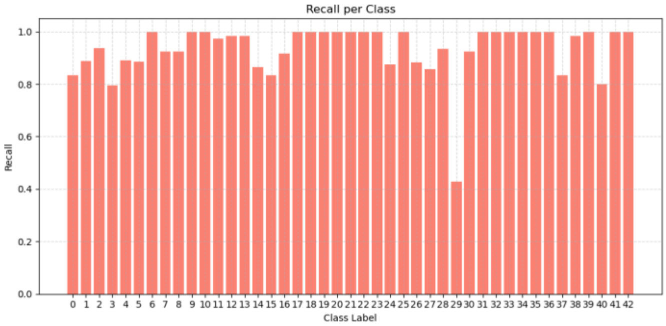


Figure 13– Histogram of Random Forest recall for each class based on test set.

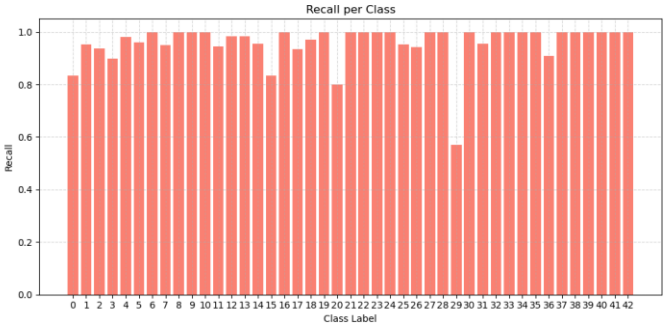


Figure 14– Histogram of CNN recall for each class based on test set.

However, Random Forest and CNN had relatively low recall for class 29 (Figure 13, 14).

We hypothesize that this low recall is due to the limited number of samples for training and testing, which is supported by the class distribution (Figure 1).

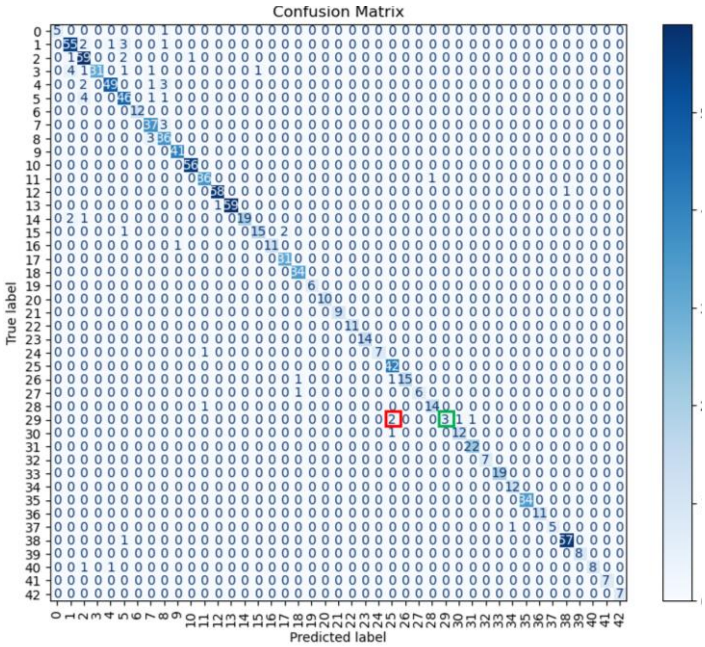


Figure 15– Confusion Matrix for Random Forest test results.

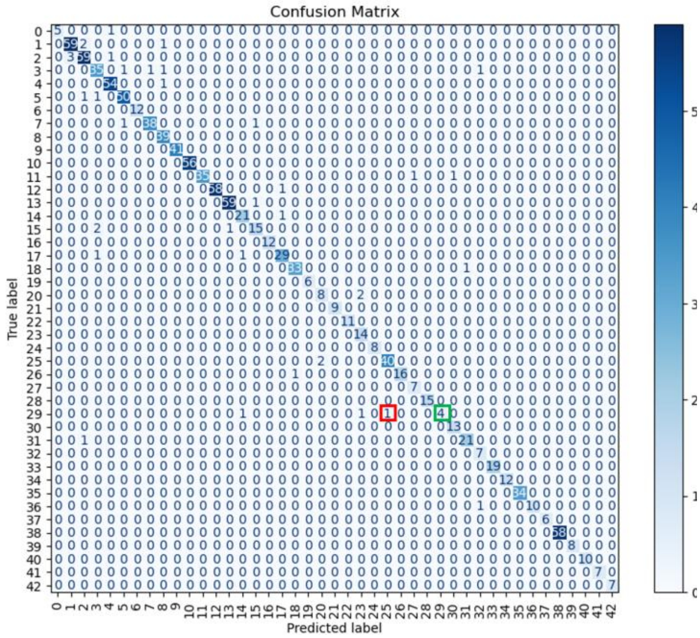


Figure 16– Confusion Matrix for CNN test results.

It appeared that both models tend to confuse class 29 with class 25 (Figure 15, 16).

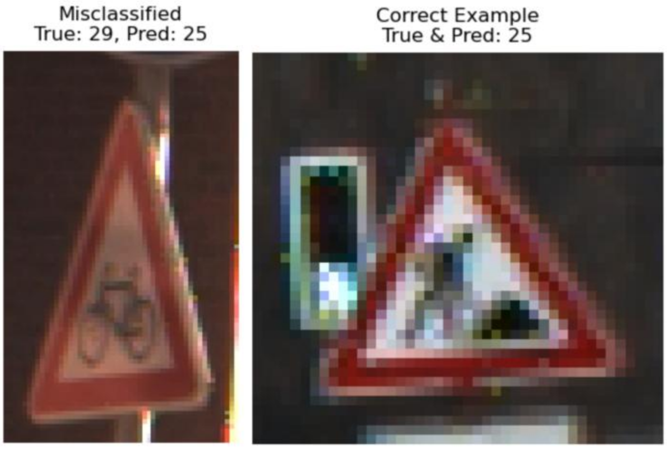


Figure 17– Example of misclassification by Random Forest & CNN for class 29. Left image ID: 229. Right image ID: 1852.

By inspecting the misclassified image (Figure 17), we hypothesize that the misclassification could also be due to the tilted viewpoint of the traffic sign.

	Random Forest	CNN
Confidence	0.17	0.4407
Percentile	2.23%	1.09%

Table 8– Confidence value and the percentile of the confidence among all predictions for Figure 17 misclassified example.

For Random Forest, we defined the confidence value as the proportion of trees voting for the predicted class. For CNN, the confidence value was defined as the highest softmax output from the softmax activation function. Both models exhibited low confidence for the misclassified image (Table 8).

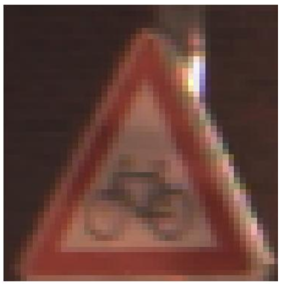


Figure 18– Perspective transformed image with simulated viewpoint. Original image ID: 229.

To verify our hypothesis, we applied a perspective transformation to the misclassified image (Figure 18).

	Random Forest	CNN
Prediction	29	25
Confidence	0.115	0.3498

Table 9– Random Forest & CNN prediction for Figure 18.

Random Forest made the correct prediction (Table 9) for the modified image, while CNN's confidence in its incorrect prediction decreased from 0.44 to 0.35. This provides evidence that viewpoint indeed affects the models' performance.

Regarding model representation, it can be inferred that HOG significantly influenced Random Forest's model, as it had a high feature importance index (Figure 6). HOG is sensitive to viewpoint variation because gradient directions change with different viewpoints. This suggests that the individual trees constructed were generally not robust to viewpoint variance.

To address this, data augmentation could be applied to diversify the images, which might improve generalization for images with different viewpoints. Additionally, more complex but viewpoint-invariant architectures could be used for CNN, such as Spatial Transformer Networks (STN) (Jaderberg et al., 2016).

5 Conclusions

In this report, we investigated various pipelines for German traffic sign classification. HOG appeared to be the most effective descriptor when used with a Random Forest classifier, highlighting the importance of edge-based structural information for sign classification. LeNet-5 CNN proved to be the most effective classifier due to its advantage of end-to-end learning. Despite strong overall performance, Random Forest and CNN's sensitivity to viewpoint variation remains a limitation.

Future improvements could include exploring more advanced CNN architectures, such as Spatial Transformer Networks or attention-based models, to enhance robustness to viewpoint changes and improve generalization. Additionally, applying data augmentation techniques and incorporating multi-scale feature extraction may further boost model performance. Finally, integrating ensemble methods that combine the strengths of both classical and deep learning approaches could offer more accurate and reliable traffic sign recognition.

5.1 Limitations

Ideally, more hyperparameters should be tuned, such as the thresholds for mutual information and PCA feature selection, the distance metric for KNN, and the maximum depth for Random Forest. The current set of candidate hyperparameters should also be expanded. However, these experiments were not conducted due to computational power limitations.

6 References

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*(6), 679–698. https://doi.org/10.1109/tpami.1986.4767851

Jaderberg, M., Simonyan, K., Zisserman, A., & Kavukcuoglu, K. (2016, February 4). *Spatial Transformer Networks*. ArXiv.org. https://doi.org/10.48550/arXiv.1506.02025

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011, July 1). *The German Traffic Sign Recognition Benchmark: A multi-class classification competition*. IEEE Xplore. https://doi.org/10.1109/IJCNN.2011.6033395

Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.