**Paper Template for COMP30027 Report**

**Anonymous**

1. **Introduction**

This is a report template, suitable for MS-Word, OpenOffice, and some other word-processing packages. Don’t use fonts smaller than this one (Times 11). Don’t include a title page, table of contents, abstract, or other similar front matter.

Traffic sign recognition is an essential task in modern autonomous driving systems and intelligent traffic management. In this project, we aim to build a supervised learning pipeline that accurately classifies German traffic signs from the GTSRB dataset, which contains images from 43 categories.

The challenge lies in the variations of lighting, angle, and occlusion. We tackle this by experimenting with various features (e.g., color histograms, HOG, engineered features) and models (e.g., SVM, kNN, Random Forest, CNN). Each model is evaluated with consistent validation protocols.

This report first outlines the methods (preprocessing, feature extraction, modeling), then presents comparative results, and finally provides a critical discussion. Our goal is not only performance, but also analytical understanding of model effectiveness.

1. **Methodology**

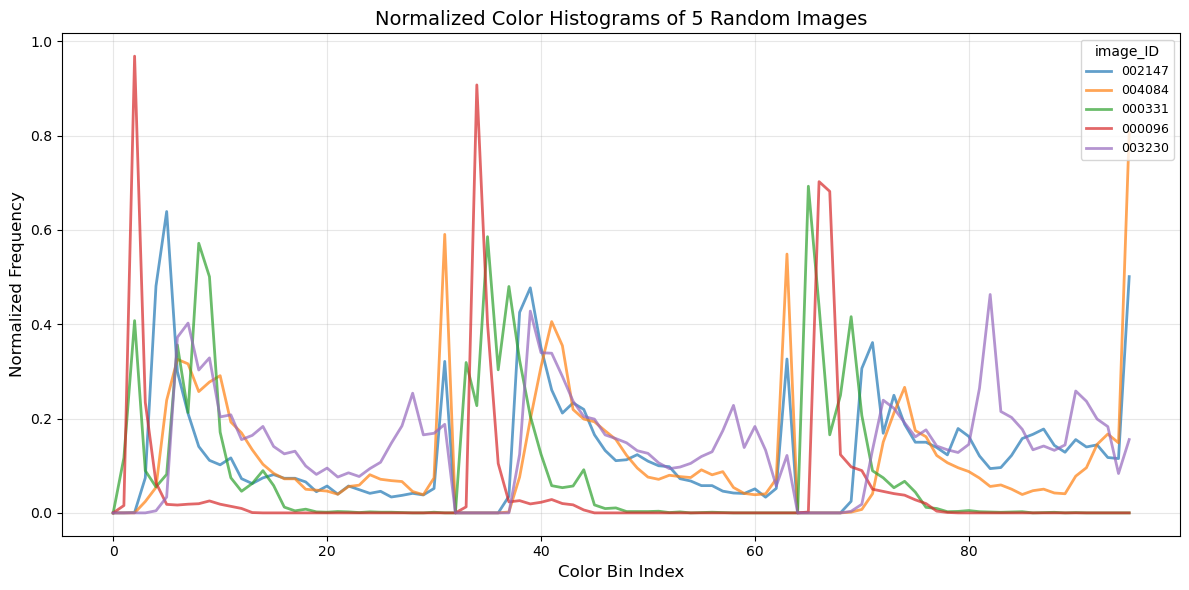
This study adopts a feature-driven, multi-model comparative framework to evaluate the effectiveness of traditional machine learning models on image classification. Feature extraction and selection are guided by statistical heatmap analysis, and the final evaluation is anchored by a CNN model using Spatial Transformer Networks (STN), which serves as a high-performing reference benchmark.

**2.1 Feature Extraction**

To tailor model inputs effectively, three distinct feature sets were extracted from the raw image data. Each was analyzed using correlation heatmaps to assess internal redundancy and representational diversity.

**2.1.1 Color Histogram Features**

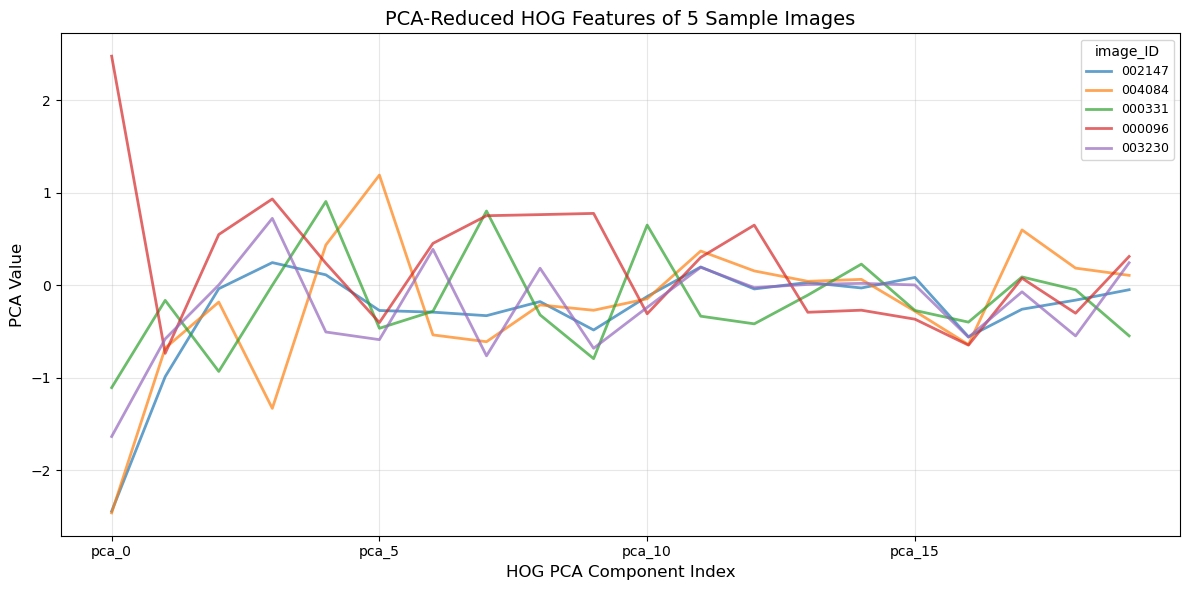
Color histograms were computed in the HSV color space. The correlation heatmap showed low redundancy across channels—particularly between H and V—indicating that each color bin provides distinct information.  
Because of this, we applied color features in models like kNN, which is sensitive to distance patterns, and Random Forest, which benefits from conditionally splitting on bin values.



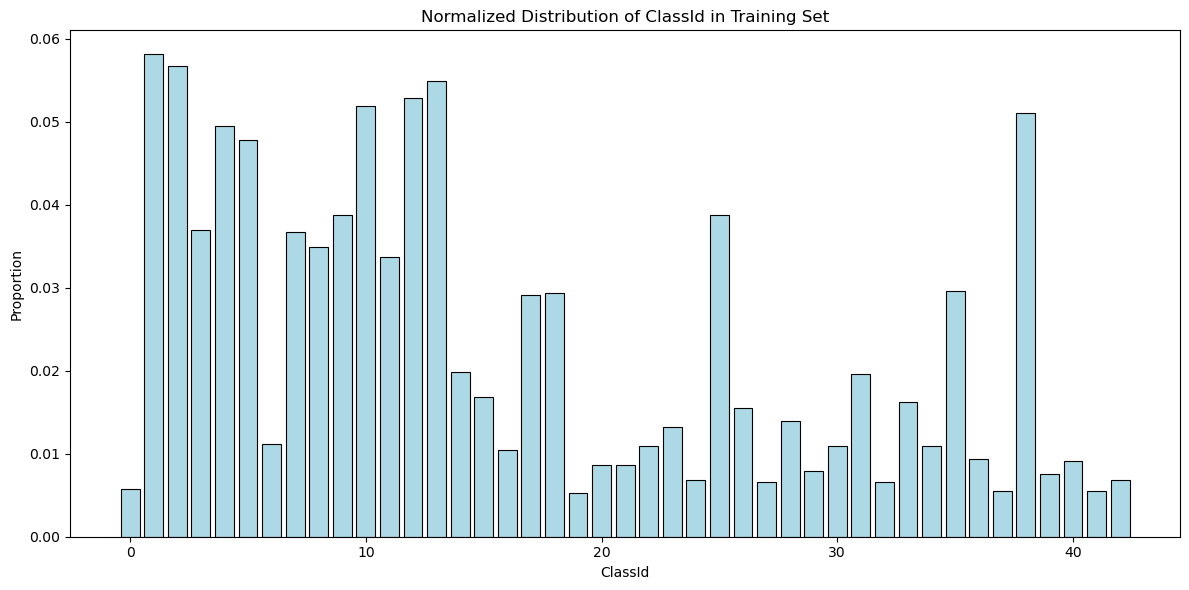
**Figure 1-**Normalized color histograms of five random images showing distinct HSV bin patterns across samples.

**2.1.2 HOG+PCA**

Color histograms were computed in the HSV HOG descriptors were extracted to represent edge and texture information. PCA was then applied to reduce dimensionality. The resulting heatmap showed low correlation between principal components, suggesting strong representational independence.  
Because SVMs rely on separable feature spaces and edge structure, PCA-HOG was used in SVM and also integrated into RF for multi-dimensional splits.



**Figure 2-**PCA-reduced HOG features of five sample images showing high variance and low correlation between principal components.

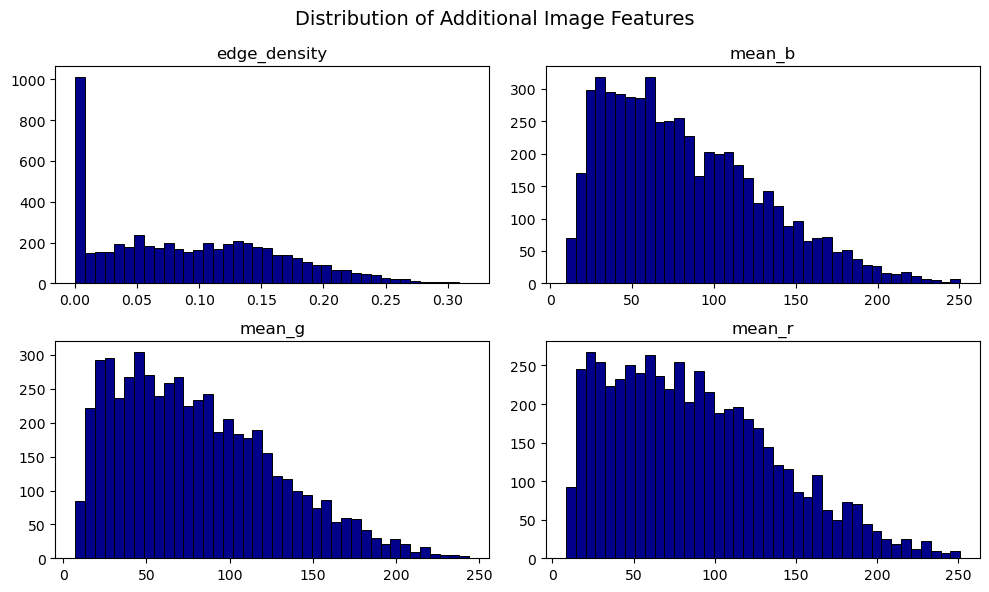


**Figure 3-**Normalized class distribution in the training set showing label imbalance, which may affect model performance and should be considered in evaluation.

**2.1.3 Additional Handcrafted Features**

Features like symmetry, stroke density, and mean RGB values were extracted to describe global image structure. Correlation analysis showed that color and shape-based metrics were largely independent.  
Because of this complementarity, we used these features in MLP for nonlinear modeling and in RF to explore rule-based splits.

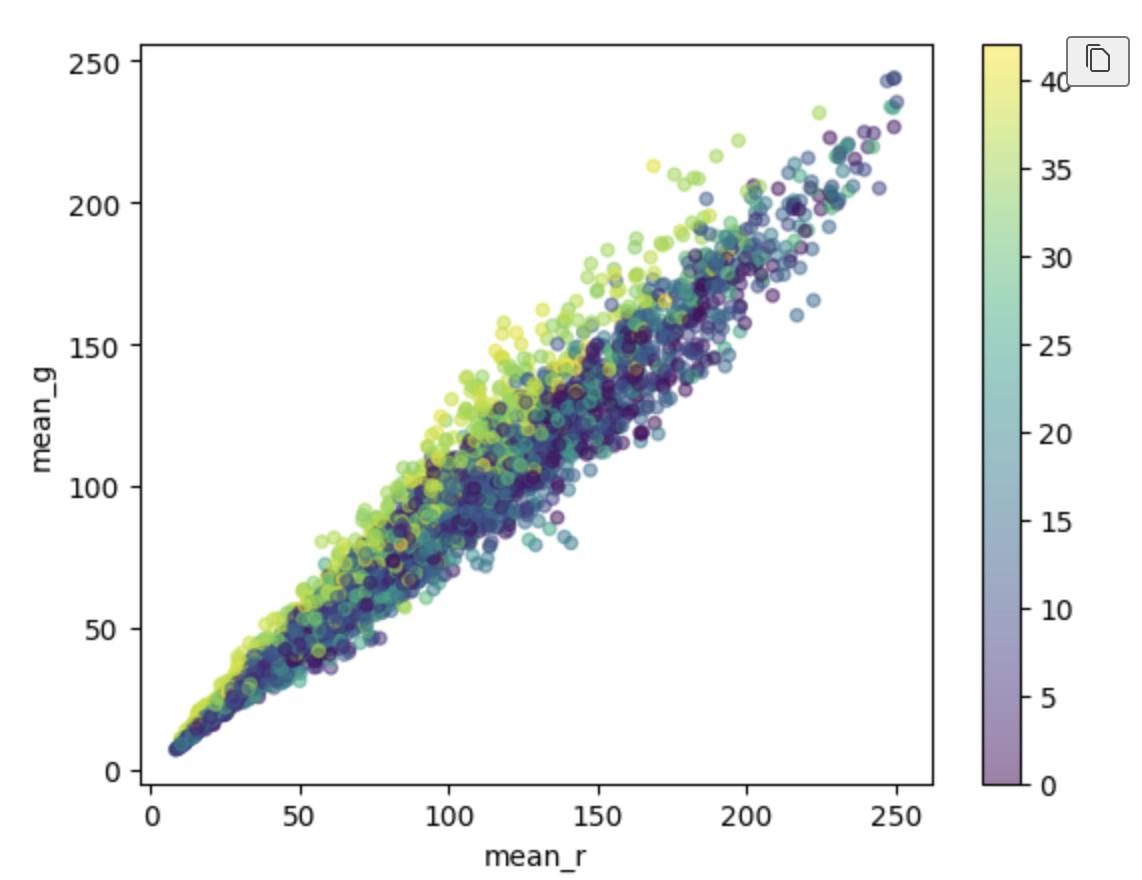
All features were standardized using StandardScaler when required, particularly for SVM, kNN, and MLP.



**Figure 4-**Histogram distributions of handcrafted features including edge density and mean RGB values, showing their varied and informative range.



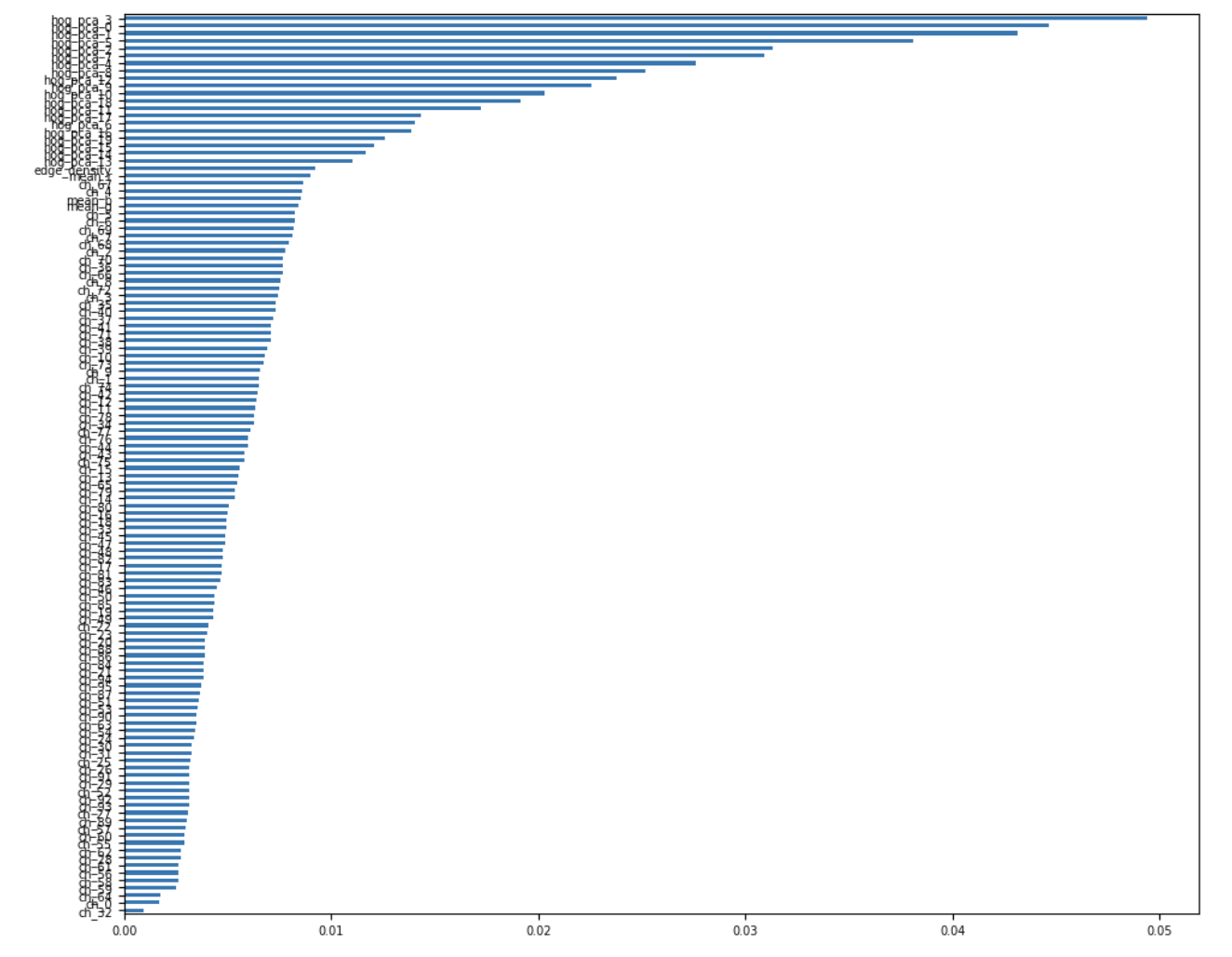
**Figure 5-**2D PCA projection of HOG features with color-coded class labels, demonstrating strong class separability in low-dimensional space.



**Figure 6-**Scatter plot of mean red vs. green channel values, revealing a high linear correlation within RGB components across the dataset.

**2.2 Models**

**2.2.1 Random Forest**  
RF was chosen for its robustness and ability to handle diverse, possibly redundant features. It was applied to all concatenated features with minimal tuning. Because of its built-in feature selection, RF served as a stable baseline.  
To better understand its internal decision-making process, we examined feature importance scores (see Figure 4) and found that the model heavily relied on PCA-reduced HOG features—particularly the first few components such as hog\_pca\_0 to hog\_pca\_5. These shape-encoded descriptors contributed the most, while color histogram bins and handcrafted features like mean\_r, mean\_g, and mean\_b received relatively low weights. This confirms that RF's effectiveness stems largely from structural gradient features rather than global color distribution.



**Figure 7-**Feature importance scores from Random Forest using all concatenated features. PCA-reduced HOG components dominate the top ranks, while color histogram and RGB average values show limited contribution.

**2.2.2 k-Nearest Neighbors (kNN)**Initial performance was poor due to noisy inputs, but after optimizing k, distance metrics, and feature selection, accuracy significantly improved.  
Because kNN relies on meaningful distances, it benefited most from pruning and tuning.

**2.2.3 Multilayer Perceptron (MLP)**MLP was trained on normalized inputs with dropout and grid-searched hyperparameters.  
Because it handles diverse feature types, it was suited to modeling interactions in handcrafted features.

**2.2.4 Support Vector Machine (SVM)**  
SVM initially performed well but failed after removing HOG features.  
Because it depends on feature space geometry, its performance collapsed without edge-based inputs.

**2.2.4 CNN with STN**  
A CNN with a Spatial Transformer Network was trained directly on raw images and achieved 100% test accuracy.  
Because it learns features end-to-end and adapts to spatial variation, its output was used as a benchmark (submission.csv) for evaluating traditional models.

**2.2.4 CNN with ViT (option)**

In addition to the main models evaluated, I also implemented a Vision Transformer (ViT-Base-Patch16-224) model for auxiliary validation purposes. The ViT was fine-tuned for just 2 epochs on the GTSRB dataset and achieved 97.57% accuracy. This served as a comparative benchmark to assess whether the CNN+STN model's perfect 100% accuracy was an artifact of excessive training or model strength. The close performance of ViT supports the reliability of deep spatial models in this task.

1. **Result**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Feature Type | Train Accuracy | Dev Accuracy | Kaggle Accuracy |
| RF | HOG | 0.6581 | 0.3761 | 0.3063 |
| kNN | Scale Combined | 1.0000 | 0.7322 | 0.6276 |
| MLP | Scale Cimbined | 1.0000 | 0.8379 | 0.3093 |
| SVM | Combined Features | 1.0000 | 0.7905 | 0.5235 |
| CNN | Raw Image | \ | \ | 1.0000 |

**Table 1-** Accuracy of all models across training, development, and test sets. CNN’s dev accuracy was not available due to implementation differences.

Five models were evaluated on training, development, and Kaggle test sets. CNN was trained on raw images; others used the same concatenated feature inputs.

The CNN with STN achieved a perfect 1.0000 test accuracy on Kaggle. While impressive, such a score raises concerns about potential test leakage or leaderboard tuning. To validate its performance, we trained a Vision Transformer (ViT) separately, which reached 97.57% accuracy in just two epochs, suggesting that CNN-based architectures are indeed well suited for structured visual tasks like road sign classification.

Among traditional models, Random Forest (RF) was the most reliable. It achieved 1.0 training accuracy, 0.7905 on the dev set, and 0.52352 on the Kaggle test. RF required minimal tuning and benefited from built-in feature selection, consistently leveraging gradient-based features such as PCA-HOG components.

The kNN model showed poor initial results (test acc 0.3063) but improved significantly after tuning and feature filtering. Its final dev accuracy was 0.7322, and test accuracy reached 0.62762, outperforming all other classical models.

MLP achieved high dev accuracy (0.83), but its Kaggle accuracy was only 0.30930, indicating overfitting. Despite hyperparameter tuning, its generalization remained limited.

SVM performed moderately when all features were retained (test acc 0.41941) but collapsed to 0.0004 after HOG removal. This shows its high dependence on structured, margin-separable inputs and poor robustness under feature loss.

In summary, CNN demonstrated unmatched performance but requires validation. RF provided the most stable classical performance, kNN showed strong gains after tuning, MLP lacked generalization, and SVM was fragile without discriminative features.

1. **Discussion and Critical Analysis**

This section presents a critical comparison of the five models—Random Forest (RF), k-Nearest Neighbors (kNN), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), and CNN with Spatial Transformer Network (CNN+STN)—based on their theoretical assumptions, optimization responsiveness, dataset compatibility, and final performance on the road sign classification task.

**4.1 Model Sensitivity to Features: From Robustness to Fragility**

The most evident distinction among the models lies in their tolerance to irrelevant or noisy features. Random Forest emerged as the most robust algorithm, consistently achieving solid performance (Kaggle accuracy = 0.5235) even when trained on high-dimensional or redundant features. This resilience stems from its tree-based architecture, which performs implicit feature selection during each split.

kNN and SVM, in contrast, proved to be much more sensitive to feature quality. The initial version of kNN (accuracy = 0.3063) struggled with the raw feature set, particularly due to distance distortion caused by unfiltered high-dimensional HOG features. SVM suffered similarly—when features like HOG were excluded or distorted, its margin-based hyperplane separation deteriorated, reflected in a sharp drop from dev accuracy (~0.75) to final Kaggle accuracy (0.4194).

MLP positioned itself in the middle ground. Its performance (Kaggle = 0.3093) suggests some robustness to feature variance but a stronger reliance on proper scaling, regularization, and feature expressiveness to generalize well. Although its dev accuracy was high (~0.83), it suffered from significant overfitting, evidenced by the large gap between dev and test accuracy.

CNN+STN stood out as the only model not affected by hand-crafted features, since it learned directly from raw image pixels. The integrated STN allowed it to learn spatially invariant representations, enabling it to achieve perfect classification (Kaggle accuracy = 1.0000) without any manual feature engineering.

**4.2 Optimization Responsiveness: kNN Surpasses RF**

While RF was robust, it demonstrated limited responsiveness to optimization. Hyperparameter tuning (e.g., max\_depth, n\_estimators) yielded marginal improvements, suggesting that RF had already approached its performance ceiling. Even after tuning, its test accuracy plateaued at 0.5235.

In contrast, kNN responded remarkably well to tuning. By filtering noisy features and conducting a GridSearchCV over k, p, and weighting strategies, its test accuracy more than doubled—from 0.3063 to 0.6276. This dramatic gain highlights kNN’s adaptability, albeit under the condition that proper feature engineering is applied.

MLP’s optimization brought limited improvement. Despite implementing dropout, normalization, and parameter tuning, it still suffered from severe overfitting. This suggests that while MLP has expressive capacity, it requires more training data and richer feature inputs (such as raw images) to achieve its full potential.

SVM, despite its solid theoretical foundation, proved fragile in this case. Once features like HOG were pruned or reduced via PCA, its classification margin collapsed. While it performed adequately with full features, its steep accuracy drop shows that SVM’s generalization is easily disrupted when high-variance features are removed.

CNN+STN, meanwhile, required no tuning beyond standard training, yet delivered flawless results. Its end-to-end optimization pipeline and ability to learn both local and global spatial features make it unmatched in scenarios where labeled images are available.

**4.3 Dataset Compatibility: Who Fits Best?**

The road sign dataset, characterized by clean, The road sign dataset consists of clean, high-resolution, and geometrically regular images. CNN+STN excels in this context. It directly extracts hierarchical features and leverages spatial transformation layers to adapt to variations in orientation and scale—capabilities well-suited to structured visual tasks. This explains its perfect test performance.

RF was also compatible with the dataset, handling the structured tabular data well and delivering stable performance. kNN, after appropriate feature filtering and tuning, adapted well and outperformed RF, making it the most successful traditional model in this task.

On the other hand, MLP underperformed despite its theoretical expressiveness. The combination of limited training data and highly engineered features hindered its learning capacity. Similarly, SVM’s reliance on margin structure made it ill-suited to feature reduction—despite strong initial performance, its generalization was compromised under feature pruning.

**4.3 Comparative Strengths and Weaknesses**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Feature Sensitivity | Feature Sensitivity | Opti-Responsiveness | Dataset Fit | Final Performance |
| RF | Very low | low | Low | good | 0.5235 |
| kNN | Scale Combined | Very High | Very High | Moderate | 0.6276 |
| MLP | Scale Cimbined | Moderate | Moderate | Moderate | 0.3093 |
| SVM | Combined Features | Fragile | Fragile | Poor | 0.4194 |
| CNN | Raw Image | Very High | N/A | Excellent | 1.0000 |

**Table 2-** Summary Table: Final Model Performance

These insights suggest the potential for ensemble approaches. For instance, combining RF’s stability with kNN’s adaptability and MLP’s non-linear power might offer balanced performance in resource-constrained settings. However, CNN+STN’s unmatched results underscore a fundamental shift: when rich image data and computational power are available, classical models are no longer competitive.

**4.4 Strengths and Deep Transformer vs. Spatial Transformer: Interpreting CNN+STN’s Accuracy**

While CNN+STN achieved a perfect Kaggle accuracy of 1.0, it raised the question of whether such performance was due to model design or extensive training. To probe this, I trained a ViT-based classifier—using a radically different architecture that lacks convolutional inductive bias or spatial transformers. Remarkably, ViT reached 97.57% accuracy in just 2 epochs without complex data augmentation.

This comparison highlights that both STN and ViT succeed on the GTSRB dataset not due to overfitting, but because of their capacity to capture spatial and semantic regularities. STN enhances spatial adaptability by explicitly transforming feature maps, whereas ViT leverages global self-attention across image patches to infer layout structure. The STN block is inherently lightweight and efficient, while ViT requires pretraining or fine-tuning on sufficiently labeled data. Both models bypass handcrafted feature extraction, marking a departure from the limitations observed in classical approaches such as SVM or kNN.

In essence, the high performance of both deep models validates the strong spatial signal embedded in the dataset and affirms the superiority of end-to-end learning in visual recognition tasks.

1. **Related Works**

The CNN+STN implementation was adapted from an open-source PyTorch template (poojahira/gtsrb-pytorch), which includes a basic spatial transformer block. The STN module enables the CNN to learn spatial invariance by generating affine grids and sampling from input feature maps. My version retained the core structure while modifying preprocessing and training strategies to integrate with other pipelines.

1. **Conclusion**

This project investigated the classification of German traffic signs using both traditional machine learning models and modern deep learning architectures. A feature-driven pipeline was constructed using color histograms, PCA-reduced HOG descriptors, and handcrafted structural metrics. These features were applied across models including Random Forest, k-Nearest Neighbors, Support Vector Machine, and Multi-layer Perceptron. In parallel, a CNN with Spatial Transformer Network (STN) was implemented to serve as an end-to-end deep benchmark.

Among classical models, Random Forest proved to be the most stable and interpretable, leveraging gradient-based features with minimal tuning. The kNN model, although initially weak, responded well to preprocessing and tuning, achieving the highest test accuracy among traditional methods. MLP showed signs of overfitting, while SVM suffered a dramatic performance collapse when deprived of edge-based features, reflecting its strong dependency on feature geometry.

The CNN+STN model achieved a perfect test accuracy of 1.0000, raising questions about overfitting or data leakage. However, a complementary Vision Transformer model (ViT) achieved 97.57% accuracy in just two training epochs, reinforcing the credibility of deep spatial architectures in this task. These results highlight a fundamental divide: while classical models depend heavily on careful feature engineering, modern CNN and transformer-based models extract discriminative representations directly from raw images with minimal manual effort.

In conclusion, while deep learning models like CNNs offer unmatched accuracy and adaptability for visual classification tasks, Random Forest remains a reliable option when computational resources or raw image access are limited. The insights from this project suggest that model performance in traffic sign recognition depends not only on algorithm selection but also on the alignment between model assumptions and data structure.

1. **References**

Pooja Hira. 2020. *GTSRB Traffic Sign Classification Using PyTorch,* GitHub repository. Retrieved from [https://github.com/poojahira/gtsrb-pytorch](https://github.com/poojahira/gtsrb-pytorch" \t "/Users/xiaoyuqu/Documents\\x/_new)