COMP30027 Report

Anonymous

1. Introduction

Classifying traffic signs, automatically identifying traffic signs from images, is a task that's growing rapidly in its usage in driver-assist and autonomous vehicle systems. In this project, I was tasked with tackling the German Traffic Sign Recognition Benchmark (GTSRB), a dataset that's publicly available and has 43 variations of German traffic signs. My goal was to explore a variety of models, from classical machine learning on hand-crafted features, to convolutional neural networks (CNNs) and multi-input deep models, and then combine them through a stacking meta-learner to maximize sign classification accuracy.

This report is structured as follows. Section 2 ("Methodology") describes the data, engineered features, and models I built conceptually. Section 3 ("Results") summarizes the performance of each approach with tables and figures. Section 4 ("Discussion and Critical Analysis") delves into why certain models succeeded or struggled, linking it with theory on feature representation, bias-variance, and ensemble learning. Finally, Section 5 ("Conclusion") details my key findings and outlines future paths forward.

2. Methodology

My pipeline for this project evolved in multiple phases:

2.1 Data & Feature Engineering

- **Raw Images:** all images used were resized to 64 x 64 pixels.
- **Provided Features:** I merged the dataset's CSV's for colour histograms (24-dim), HOG-PCA (20 principal components), edge density, and mean RGB values, yielding 120 numeric features.
- **Additional Features:** I later extracted Local Binary Pattern (LBP) histograms at multiple scales (1, 2, 3) and methods (uniform & rotation-invariant), as well as per-channel LBP in the RGB space, and concatenated these with the original features for the enriched "hand-crafted" feature set.

2.2 Baseline Models on Given Features

- **k-Nearest Neighbours** (k = 5)
- **Random Forest** (100 trees, balanced class weights)

  Each of these was wrapped in a StandardScaler -> Classifier pipeline and evaluated via stratified 5-fold CV on accuracy, precision macro, and recall macro, where the Random Forest classifier outperformed on all three metrics.

2.3 Hyperparameter Tuning

- A RandomizedSearchCV was run on the Random Forest classifier to search on hyperparameters, exploring number of trees, tree depth, maximum features, and leaf parameters. This totalled 250 fits.
- Using the best parameters provided by this search, I was able to boost the Random Forest accuracy from ~79.2% to ~81.2%.

2.4 Deep Learning on Raw Images

- **CNN From-scratch Trained on Images:** a small 3-conv-layer network (filters 32-128), trained with sparse categorical crossentropy and on-the-fly augmentations. This achieved a ~91.8% accuracy on a local 80/20 train/validation split. Further experimenting with different activation methods and layers led to a local validation set accuracy of ~97.7%. (Goodfellow, Bengio & Courville (2016))
- **MLP Model Trained on Features**: This was a three-layer MLP model trained exclusively on all the features provided in the dataset CSVs. This yielded a ~79.9% accuracy on a local validation split.

2.5 LBP + Gradient Boosting

- Extracted multi-scale and colour-aware LBP features.
- Trained a HistGradientBoostingClassifier (Friedman (2001)) on these. Using a RandomizedSearchCV on this to tune hyperparameters totalled 150 fits and gave the best parameters, which when used, led to an accuracy score of ~44.8% on a local validation split of the training dataset.

2.6 Stacking Meta Learner

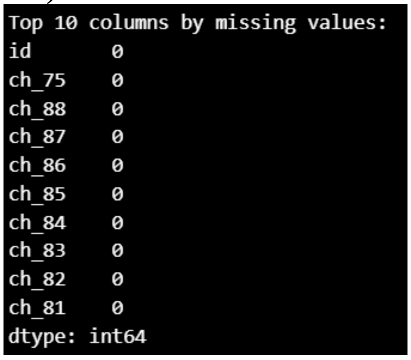
- Generated out-of-fold softmax probabilities for the three base models (CNN, MLP, LBP-GB) using a fold held out during training.
- Stacked all of these values in one array.
- Trained a logistic-regression meta-model (multinomial, saga solver) on the OOF Data.
- The final stacked accuracy on the held out validation set reached ~98.5%, the higher than any single models so far.

3. Section

This section will summarize the performance of each approach with tables and figures.

3.1 Exploratory Data Analysis (EDA)

EDA showed no missing values in the data (Figure 1) but uncovered an uneven distribution of classes in the dataset (Figure 2).



```
Top 10 columns by missing values:
id       0
ch_75    0
ch_88    0
ch_87    0
ch_86    0
ch_85    0
ch_84    0
ch_83    0
ch_82    0
ch_81    0
dtype: int64
```

Figure 1 - Code output showing columns in order of missing values and number of missing values.

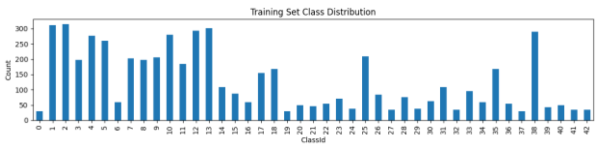


Figure 2 – Bar graph showing the amount of samples present in the dataset for each traffic sign type.

3.2 Baseline & Tuned Classical Models

Random Forest (RF) classifier outperforms kNN on all three metrics. (Table 1)

Model	Accuracy	Precision Macro	Recall Macro
kNN	61.6%	66.9%	56.3%
RF	79.2%	86.3%	73.5%
RF (tuned)	81.2%	88.0%	76.9%

Table 1- Metrics comparing the performance between all three baseline and tuned models.

3.3 Deep Models

The CNN model based on the raw image data provides better accuracy than the MLP model based on given features or the LBP model based on extracted textures and gradients. (Table 2).

Model	Test Accuracy	Validation Accuracy
CNN	94.7%	97.7%
MLP	74.6%	79.9%
LBP (tuned)	42.3%	44.8%

Table 2- Metrics comparing the accuracy between the CNN, MLP, & LBP models on training data as well as a local validation set.

3.4 Final Stacked Ensemble/Meta Learner

The first attempt at the ensemble model included using a fixed alpha value which specified how much to weigh in predictions from each model. After creating the LBP, I used a logistic regression model that would automatically learn the best weights to apply to each prediction.

Stacking Method	Test Accuracy	Validation Accuracy
Fixed α = 0.6	93.4%	95.9%
3-model stack + logistic regression	95.8%	98.5%

Table 3- Metrics comparing the accuracy between the different stacking methods on validation sets as well as the test set, as measured on Kaggle.

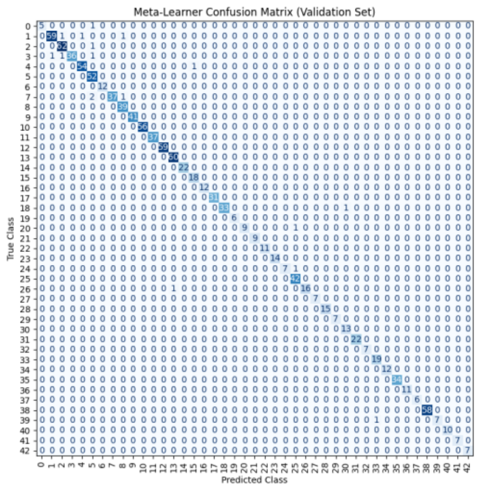


Figure 3- Confusion matrix showing the results of the meta-learner model on a local validation set

4. Discussion and Critical Analysis

4.1 Feature Representation & Bias Variance

The features included in the dataset (colour histogram, HOG-PCA) provided a strong low-variance signal, which was used effectively by classical learners such as Random Forest but plateaued after a certain point. The bias in this instance most likely stemmed from the limited information carried in PCA-reduced HOGs vs the full texture/shape utilized by the CNN. The CNN gained special and edge patterns directly from the images, drastically reducing bias and achieving much higher accuracy. This comes at the cost of higher variance, being sensitive to hyperparameters and augmentations. The MLP model on the other hand, learned from the feature data already present in the dataset. This ensured that the provided future data would not go without purpose, and allowed the MLP to learn what the CNN couldn't from solely images.

4.2 LBP Branch Analysis

Using pure LBP was too local and did not account for colours, which led to an extremely high bias and very low accuracy. To combat this, a multi-scale and colour LBP was added so that the model could not only capture micro textures but also colour distribution. When concatenated with the original information present in the dataset, this provides a considerable improvement over a simple LBP model. However, this LBP histogram method remains sensitive to rotation and light, one of which can be fixed using rotation invariant LBP, but that will drastically increase the compute time. (Ojala, Pietikäinen & Mäenpää (2002))

4.3 Ensemble & Stacking Theory

The stacked meta-learner was used to leverage the respective strengths of all three models. While the CNN excelled on geometry and texture patterns, the MLP was able to add information related to global colour and lighting, and the LBP added self-extracted histograms. Using logistic regression, the model learned class-specific weightings of each branch's confidence, allowing it to reduce variance and bias in the ensemble model. Calibrating the probabilities from each model further could sharpen confidence values, refining the meta-learner's input distribution.

4.4 Error Analysis

The confusion matrix (Figure 3) shows a small number of misclassifications amongst visually similar signs, such as those regarding speed limits. Adding more shape-specific augmentations may help overcome this further. The rarest classes, with the fewest training examples, will still have lower recall. To fix this issue, oversampling or adding to the dataset via generative augmentation may help.

5. Conclusion

Demonstrated in this project is a comprehensive exploration of multiple classical and deep learning methods for traffic sign-classification. Starting with simple kNN and Random Forest implementations on given image features, I stepped further and showed how training on the images directly (CNN) can improve accuracy, and how using given image features (MLP) alongside new extracted patterns (LBP) can provide a far better representation and allow the model more information to predict a certain class. Ultimately, all this information was put together through a stacked meta-learner, which learned how to weigh the inputs from each of the three models most effectively, allowing for further improvements in performance.

My key takeaways from this project include:

- **Feature diversity** is powerful. Combining multiple types of features into different models that work best with them utilizes unique strengths and allows for a more complete picture.

- Stacking **OOF** (out-of-fold) predictions robustly reduces the variances or biases present in any single learner.
- A **critical analysis** of error patterns can help guide targeted improvements, such as specialized augmentations or oversampling for rare classes.

In future work on this project, I'd be interested in applying transfer learning, exploring sophisticated data augmentation, synthesizing data, and streamlining the model to be able to run in real time.

6. References

Ojala, T., Pietikainen, M., & Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7), 971–987. doi:10.1109/TPAMI.2002.1017623

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. The Annals of Statistics, 29(5), 1189–1232. http://www.jstor.org/stable/2699986

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.