Name That Traffic Sign

Yasin Kip

1 Introduction

In this report, we train several machine learning models for the sake of traffic sign recognition. Our models are trained on a dataset consisting of images of traffic signs and respective labels classifying them on the basis of their functionality. Features were extracted from the raw images using image recognition techniques, and these features were then used to predict the class label of unseen test data. The best performing model out of those we've chosen to implement (with respect to the evaluation metrics used) was found to be the Random Forest model, which correctly predicted the type of traffic sign with an accuracy of about 80% on unseen data.

2 Methodology

2.1 Feature Extraction

Feature extraction was a big part of this task. The first features we made an attempt to classify was the shape, color and text on each traffic sign.

2.1.1 Shape detection

To extract the shapes from the image, we first preprocessed the image by Gaussian blurring then employed Canny edge detection, which is a technique used to identify the edges in an image.

Figure 1: Canny output on training sample

After obtaining the edge image, we attempted to classify shapes by fitting contours and comparing the output to a sample of 'ideal' shapes, and whichever template the image had the greatest similarity with was determined to be it's shape.

2.1.2 Color detection

For color classification, we performed color segmentation on each image. In other words, we filtered out all color from the image which did not meet certain requirements. These requirements were constructed in the HSV color space, which is more practical than RGB values in this case because the continuum of colors a human would classify as 'red' (and similarly for other colors) can lie in a single interval in the HSV spectrum, contrary to RGB. It was important (see Figure 2) to employ a range of preprocessing steps on the input images, including brightness normalisation and border removal, the latter of which is based on the realisation that most traffic signs were centered in each image.

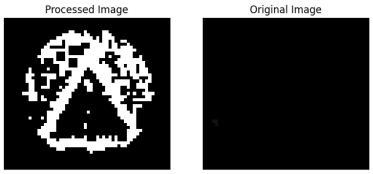

Figure 2: Color segmentation result before and after processing

For each image, we applied color segmentation on three of the four possible colors red, yellow and blue then took the contour (continuous line) drawn with the largest area on all three segmented images. The color used to segment the image which contained the continuous contour of largest area was then classified as the color of the traffic sign in the image. The confidence used for this color prediction was given by

$$\text{Conf} = \frac{\text{Area}(C_1)}{\text{Area}(C_2)\text{Num}_1} \tag{1}$$

Where C_1 is the area of the largest contour for the 'best' color, C_2 is the area of the largest

contour for the second 'best' color, and Num_1 is the total number of contours found in the 'best' color segmented image. The Num_1 term is included to punish the confidence rating for color segmentations whose contours were being fit to noise in the image. The last color (gray) was chosen if the confidence for the color prediction was below a certain threshold.

2.1.3 Detecting Fill

We used the setup in the color detection phase to detect whether or not the traffic sign was "filled" with color or not (like a "STOP" sign and unlike a speed sign). This was simply done by taking the best contour found in the previous section and checking how many pixels inside of it were filled, as a proportion. If this proportion passed a threshold, it would be considered filled. See Figure 3.

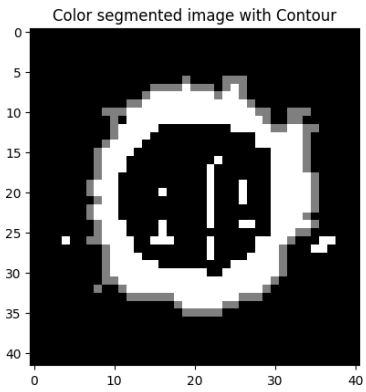

Figure 3: Area bounded by the gray contour is 59% filled

2.1.4 Detecting Text

A variety of text detecting packages were trialled, such as easyOCR and tesseract, but they produced remarkably poor results due to the low quality, blurry text on the traffic signs. In the end, we opted for the training of a simple convolutional neural network (CNN) for traffic sign detection. We hand-labelled the 10 classes which had text on them, then trained a CNN on the 10 possible text labels.

2.1.5 Other

Other feature extractions include HSV histogram channeling and Local Binary Pattern (LBP) with PCA, which is a measure of texture in the image. Given with the dataset was the mean RGB, RGB color histograms, and histogram oriented gradient (HOG) with PCA.

2.2 Model Fitting

2.2.1 Encoding and Standardisation

Features such as color were encoded using one-hot encoding for use in the model fitting phase. Because we were fitting an SVM, we standardised our data as much as possible. Some heavily skewed continuous features such as color histogram needed to first be transformed before they were standardised.

2.2.2 Hyperparameter and Feature selection

The models fit were decision trees, support vector machines (SVMs), and random forest models. Mutual information was used to rank feature importance and K-fold cross validation with grid search was used to choose the hyperparameters of each model, as well as the top k features to keep. Hyperparameters chosen by K-fold (K=5) cross validation include the max depth and splitting decision (best vs random) for the decision tree, the slack parameter and kernel function for the SVM, and the number of estimators for the random forest model.

3 Results

Our models were evaluated using confusion matrices and classification reports for all 43 classes. Overall accuracy and weighted average f1-score were the main evaluators used to evaluate model performance. The models were then tested for accuracy on a kaggle test dataset, for which no evaluation metrics are available except accuracy.

3.1 Decision Tree

The decision tree had an overall accuracy of 72% on the validation set, a weighted average f1-score of 0.72, and a training accuracy of 99%. It had a max depth of 30, picked the 100 best features (out of 311), and the splitting was not done randomly. On the test set, the model had 56% accuracy. Figure 4 depicts the confusion matrix of the decision tree and Table 1 lists the top 5 features.

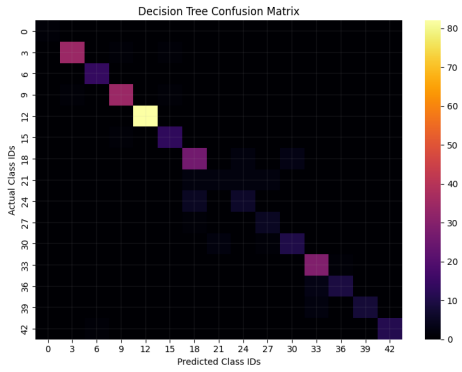

Figure 4: Confusion matrix of decision tree

Feature	Importance
color blue	7.1
ch 59	5.9
ch 26	5.8
ch 63	5.6
s 15	5.3

Table 1: Relative feature importance top 5 of decision tree. 'ch' stands for color histogram, and 's' for saturation.

3.2 Support Vector Machine

Our SVM model had an overall accuracy of 85% on the validation set, with a weighted average f1-score of 0.85, and a training accuracy of 100%. A slack parameter of $C = 1$ was chosen with a linear kernel. Feature selection was not performed for this model, since it performs well on higher dimensional data. On the test set, an accuracy of 74% was achieved. The confusion matrix of the SVM is depicted in Figure 5.

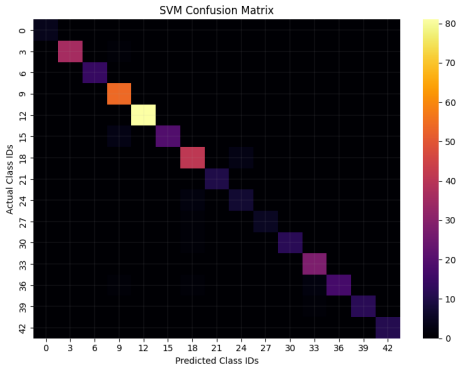

Figure 5: Confusion matrix of SVM

3.3 Random Forest

Our random forest model also had a training accuracy of 100%, but a slightly lower validation set accuracy of 82%, with a weighted average f1-score of 0.82. The random forest model was built on 200 estimators, and only the top 150 features were selected on it's training. Despite the lower validation set accuracy, the random forest model outperformed the SVM on the test data, scoring 80% accuracy. See Figure 6 and Table table2 for the confusion matrix and feature importances respectively.

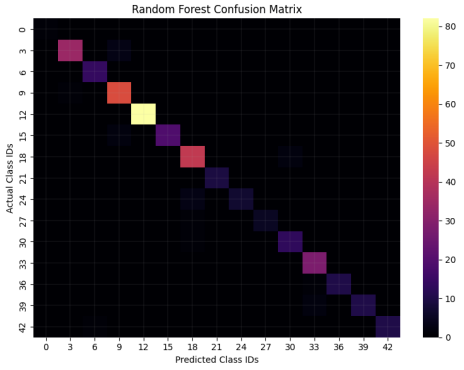

Figure 6: Confusion matrix of random forest

Feature	Importance
color blue	3.6
v 40	3.6
hog pca 4	3.4
ch 67	3.4
s 83	3.1

Table 2: Relative feature importance top 5 of random forest

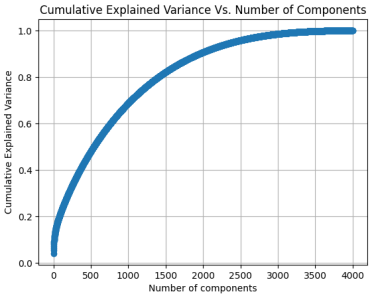

Figure 8: PCA cumulative explained variance ratio curve on LBP

4 Discussion

4.1 Feature extraction methods

Some feature extraction methods such as shape and fill detection ended up not being accurate enough to rely on, as they would only introduce noise to our data. The shape detection method failed because contours could only be drawn on continuous lines; if even a single pixel was missing, contours which would otherwise be perfect triangles are no longer even detected (see Figure 7). A possible solution was morphing, but that amplified the noise in noisy images by far too much.

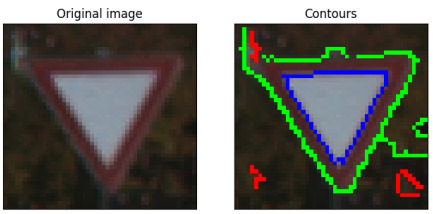

Figure 7: Green contours were identified as circles, blue as triangles, red as quadrilaterals.

Our fill detection methods demonstrated an accuracy of about 80%, which hurt the performance of many models, so it wasn't a feature we used. The method used would overestimate the fill ratio of noisy images with an overall red or blue hue. In the future, one should first normalise for the average hue in each image. We also couldn't use the LBP output with PCA, as it would take over 1500 PCA components for the cumulative explained variance ratio (see see Figure 8) to exceed 0.8, which multiplied the dimensionality of our data by 5 and caused overfitting in many models.

Our color and text detection were accurate about 90-95% of the time, and they ended up being useful features, particularly in the decision tree and random forest models, as per Table 1 and Table 2.

4.2 Model performance

Looking at the training accuracy versus validation set accuracy versus test set accuracy in all three of our models, there is a glaring issue. Each model had very high accuracy in the training set, lower accuracy in the validation set, and even lower accuracy in the test set.

The first issue can be explained by overfitting; our models have a very high variance and low bias. This means that none of our models were too "simple" to capture the trends in our data. Rather, they stuck too closely to the trends and were unable to generalise well. In the decision tree, various parameters can affect the models tendency to overfit: in our case, we tested a handful of parameters using K-fold cross validation, but this testing is in no way exhaustive. The larger we make our parameter grid, the longer the computation time. The same can be said for the SVM and the random forest. Other parameters need to be tested more thoroughly.

The second issue is slightly more nuanced. Following from the first issue, overfitting is definitely present. Since the training set is larger in this phase, there is even more tendency to overfit. For example, in the first phase, where we are using only the training sample and splitting into 80/20 train and test then choosing hyperparameters by K-fold cross validation on the 80%, our SVM model yielded a linear kernel with a slack parameter of 1. When training the model on the entire dataset however, K-fold cross validation instead yielded a slack parameter of 10 and a radial basis function (rbf) as the kernel. A similar pattern was observed with the decision tree. These hyperparameters are more prone to overfitting then the previous, which would then manifest in an even lower accuracy on the test set. This reasoning is consistent with the fact that the random forest was the only model whose validation set accuracy was not much dif-

ferent to it's test set accuracy: the hyperparameter we chose by K-fold cross validation did not have a strong effect on it's overfitting habits, so there was no large discrepancy in overfitting between the training and test set phases.

4.3 Error Analysis

Our models' confusion matrices suggest that they all made mistakes in the same areas. This points to a possible shortcoming in feature extraction. The images which were hardest to classify were those with class id's in the 21-30 range, often being misclassified as those in the 15-18 range (see Figure 9). There are also confusions within the 21-30 range (see Figure 10).

Figure 9: Comparing signs with class id equal to 16 and 23

Figure 10: Comparing signs with class id equal to 21 and 24

Our models would likely be able to distinguish between those in the 15-18 range from the 21-30 range if we could classify shape correctly, as evidenced by Figure 9, but there were no features considered that would be effective at differentiating the two signs in Figure 10. General features like color histograms and HOG picked up some patterns, but they evidently were not enough to reliably differentiate between the shapes inside each sign. As a result, none of our models were able to learn how to accurately tell apart signs with distinct inner shapings, like these.

5 Conclusion

A large challenge in this problem is extracting features from the images with various levels of sharpness and quality. Simple image recognition techniques fail in general. A more robust solution to feature extraction and indeed the problem of sign recognition as a whole is a more involved employment of CNNs, which can reach an accuracy of 99.2%[1]. The models used in this investigation, however, could benefit from a more exhaustive search over their parameter spaces, particularly in hyperparameters managing the tolerance of overfitting, as it greatly impacts each models performance.

References

Md Nafiujjaman Alam, Md Hasanul Islam, Md Al Mehedi Hossain, Md Saiful Islam Uddin, Md Badrul Hasan, and M Shamim Akhtar. 2022. A hybrid deep learning model for traffic sign recognition using cnn and svm. *Heliyon*, 8(12):e12088.

[1](Alam et al., 2022)