Sean Sungil Kim

March 19, 2019

GTSRB Traffic Sign Image Multi-class classification

Abstract

The purpose of Computer Vision would include but not only be limited to image detection and recognition. For example, successfully classifying traffic signs given image data of traffic signs could benefit vehicle companies tremendously. To be able to classify traffic signs with significant performance, finding the optimal model with corresponding hyperparameters is important. Therefore, implementing machine learning algorithms in addition to analytical algorithms is necessary. The goal of this traffic sign image multi-class classification analysis is to explore the effectiveness of different classifiers on lower quality (both in dimension size and color), non-centered traffic sign image (researched and provided by INI Benchmark, which can be downloaded here: http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset). Being able to build models with powerful predictive performance using lower quality and non-centered image data can excel in environments that are extremely time sensitive and limited in computing power, which are very frequent in the real world. Random Forest and Multi-Layer Perceptron classifiers fit on wrapper-based feature selected images demonstrated significant testing f-score performance of 0.879842 and 0.911308 respectively. The model fitting and the hyper-tuning steps also required extremely less time compared to Gradient Boosting, Ada Boost, SVM and XGBoost. Random Forest had the fastest runtime with lower f-score performance, whereas Multi-layer Perceptron took significantly longer than Random Forest but had the highest f-score performance among all models explored.

Introduction

Image data are heavy both in terms of size and processing time required. It is also difficult to obtain carefully centered image data (regardless of what the images are of) in the real world. Having high quality, colored and perfectly centered images can enhance the effectiveness of machine learning and statistical algorithms. However, this requires a significant amount of time, both in the front and the back end. Utilizing lower quality image data is extremely beneficial when computing power is very limited. In addition, being able to use not-so-perfectly centered images can reduce time costs. This reduction in quality can be extremely beneficial, if these resources are scarce.

The data used for this project is from INI Benchmark. INI provides data for computer vision and machine learning problems. The traffic sign data utilized is named GTSRB (German Traffic Sign Recognition Benchmark), which has total 39,209 images and 43 classes (traffic sign type) in the training set alone. The data comes with the actual images and annotations, in addition to pre-computed features (such as HOG and haar-like). All of these were not utilized in this project. Everything was started from scratch. The image sizes vary between 15x15 and 250x250 pixels. The images are not always squared and/or centered, which explains why INI Benchmark provides the location of the actual traffic signs in a bounding box format for each image in the annotation csv file. Since the goal of this project is to classify low quality traffic signs, all images were converted to grayscale and resized (downsized) to 16x16 pixels. Because of the original size inconsistency, some images with bigger original dimensions may have more

detailed information than the smaller images post downsizing. One downsized grayscale image from each class are portrayed below in Figure 1. The comparison of original vs. information-lost images from converting to grayscale and downsizing is illustrated below in Figure 2.

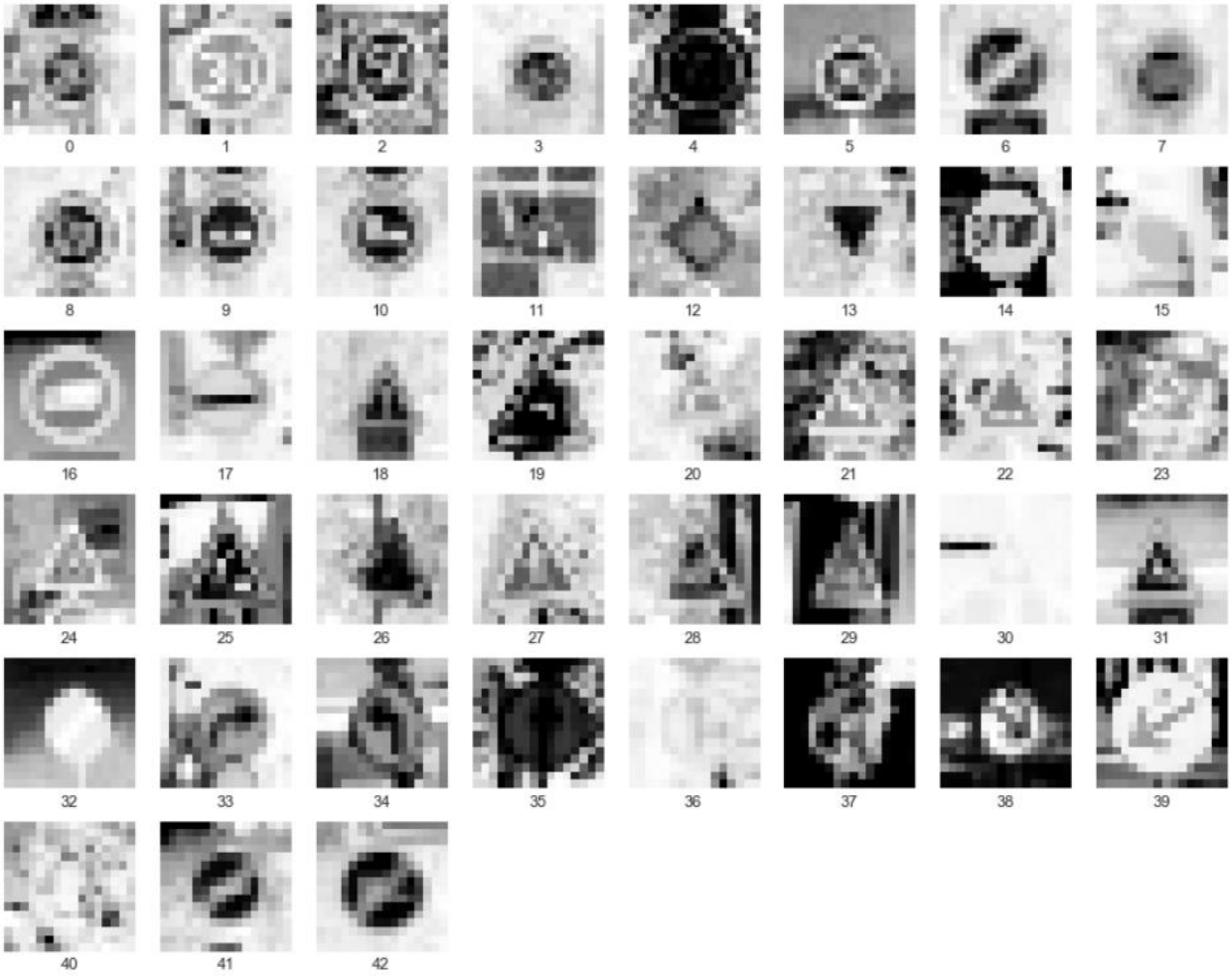


Figure 1: One downsized grayscale image from each class, total 43 classes.

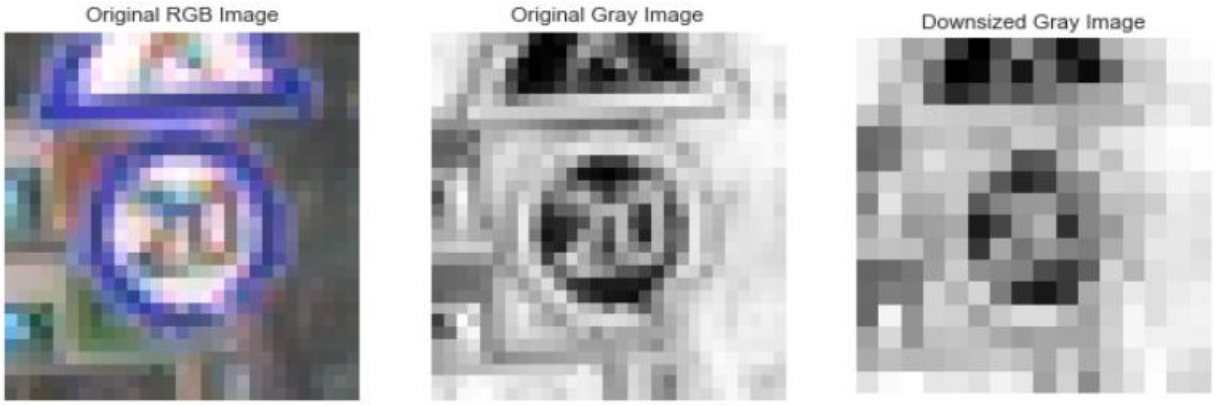


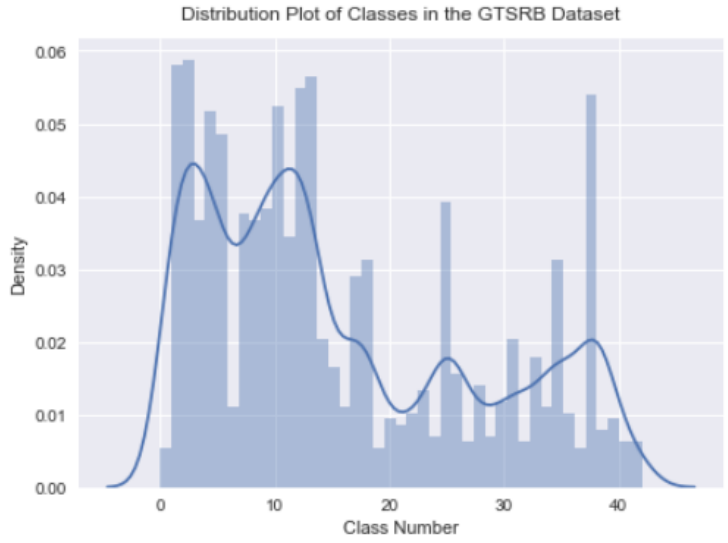
Figure 2: Comparison of original RGB vs. grayscale vs. downsized grayscale images.

Literature Review

The state-of-the-art traffic sign recognition research proved that tree-based ensembled methods (such as Random Forest [1], [2]), Convolutional Neural Networks [3], [4], [5] and SVM [4] classifiers excel with high performance. The non-network classifiers such as Random Forest and SVM usually go through a learning phase on feature extracted/selected data, such as Histogram of Oriented Gradients (HOG). This is to remove the clutter within the training images and only focus on the detected high contrast edges of the traffic signs [2], [4]. Convolutional Neural Networks go through this feature extraction stage in a way as well, because the convolution layer acts as a learning layer that extracts important pixel patterns, or in this case, the traffic signs within the image. These CNN models have multiple other hidden layers as well, such as max-pooling, which are crucial for invariant object recognition [3], [4]. Since the goal of these researches is to obtain the best performing models, these models learn on large datasets, with image dimensions big enough to capture all necessary information. Their parameters are hyper-tuned thoroughly, investing a lot of time costs. Although the goal of this project is a bit different from that of these published researches, the proposed methods provided a general starting point. Similar to the researches referenced, dimension reduction, feature extraction, and feature selection methods were explored in this project with the hopes of increasing performance. Moreover, due to the significance of Random Forest, Neural Networks, and SVM classifiers, they were included in the list of classifiers that were examined in the exploratory model fitting phase.

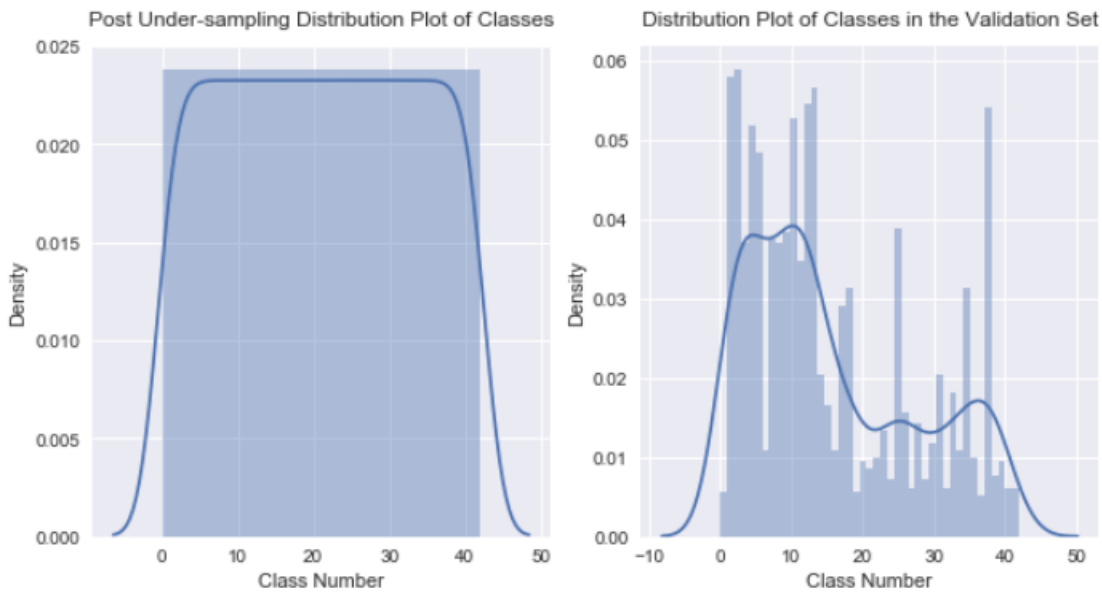
Methodology

First, the distribution of the class labels was analyzed. The original data had class imbalance issue, which is illustrated in Figure 3 below. Since computing power and time were very limited, random under-sampling was performed. The lowest count of the unique class labels was computed, and this was set as the max number of each class count for the under-sampling procedure. Instances from each class were chosen randomly, without replacement. This resolved the class imbalance problem and reduced the dataset size (from 39,209 to 10,670 images) as well.




```
Total 39209 images in the GTSRB dataset
Total 39209 classes in the GTSRB class
```

Figure 3: Original class distribution of GTSRB data.




```
Input data dimensions: (8514, 256)
Target data dimenstions: (8514, 1)
79.79 percent training set, 20.21 percent validation set
```

Figure 4: Comparison of the training vs. testing class distribution post random under-sampling.

Training set was kept at roughly 80% and the validation set was kept at roughly 20% post under-sampling. The validation set, however, was left imbalanced to evaluate the true model performance on the data with original integrity. The balanced training vs. unbalanced testing class distribution graph and the split ratio are shown in Figure 4 above.

After the preprocessing step, the downsized grayscale images were first explored with a handful of different classifiers to analyze their base performance (models with default parameters). A list of classifiers reflected on the references were included, such as Random Forest [1], [2], Convolutional Neural Networks [3], [4], [5] and SVM [4]. For comparison, other widely used boosting ensemble algorithms such as Gradient Boosting, Ada Boost and XGBoost were included in the list as well. All exploratory model comparisons were performed using default parameters. In the model performance output using 16x16 grayscale images in Figure 5 below, Gradient Boosting, Multi-layer Perceptron and XGBoost demonstrated cross-validation accuracies of over 84%. Among these classifiers, Gradient Boosting and XGBoost had extremely long CV runtime. Gradient Boosting took almost twice as long compared to XGBoost. In addition to the shorter runtime compared to some of these boosting methods, Multi-layer Perceptron Classifier had the highest CV accuracy of 88.5%. Considering the computing power and time required, only the Multi-layer Perceptron model was regarded as significant from this output.

```
Comparing Different Classifiers:
------------------------------------------------------------------
Random Forest Classifier Acc: 0.753 (+/- 0.01)
CV Runtime: 11.713051319122314
------------------------------------------------------------------
Gradient Boosting Classifier Acc: 0.840 (+/- 0.01)
CV Runtime: 7309.67308306694
------------------------------------------------------------------
Ada Boost Classifier Acc: 0.179 (+/- 0.09)
CV Runtime: 147.57923245429993
------------------------------------------------------------------
SVM Classifier Acc: 0.330 (+/- 0.02)
CV Runtime: 522.8464555740356
------------------------------------------------------------------
Multi-layer Perceptron Classifier Acc: 0.885 (+/- 0.01)
CV Runtime: 455.1103262901306
------------------------------------------------------------------
XGBoost Classifier Acc: 0.845 (+/- 0.02)
CV Runtime: 3865.54155421257
------------------------------------------------------------------
The classifier (with default parameters) with the highest accuracy is
Multi-layer Perceptron Classifier
The cross-validation accuracy is 0.885 (+/- 0.01)
```

Figure 5: Exploratory model comparison output using 16x16 grayscale traffic signs.

After the exploratory model comparisons using 16x16 grayscale images, thresholding was performed on the data to remove background clutter in the images as a feature extraction approach. 5 different thresholding algorithms were compared visually, shown in Figure 6 below. Global and Otsu's thresholding methods captured the traffic sign images favorably for some instances but eliminated the entire traffic signs for others. Otsu's with Gaussian Filtering approach did not quite work for a lot of images, so this was not considered. Adaptive mean thresholding was chosen to be applied on the data, since a lot of the observed traffic signs were readable by human eye with the least clutter.

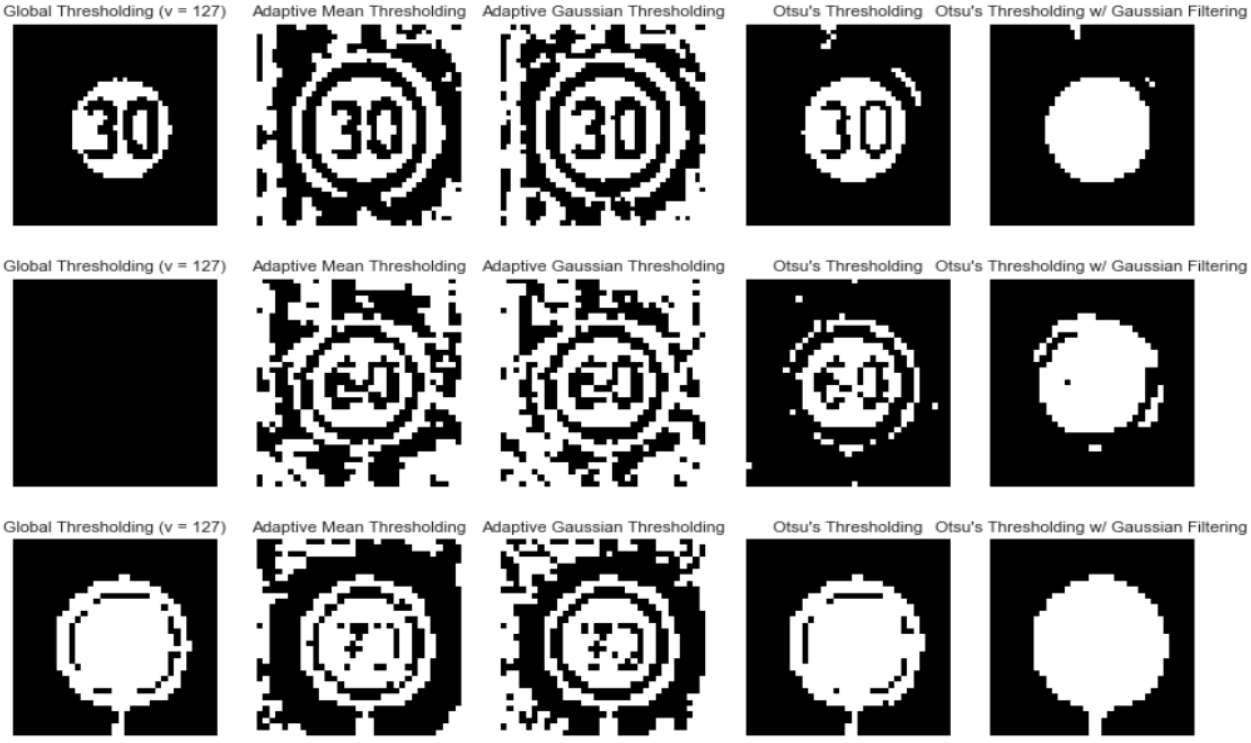


Figure 6: Comparisons of 5 different thresholding methods.

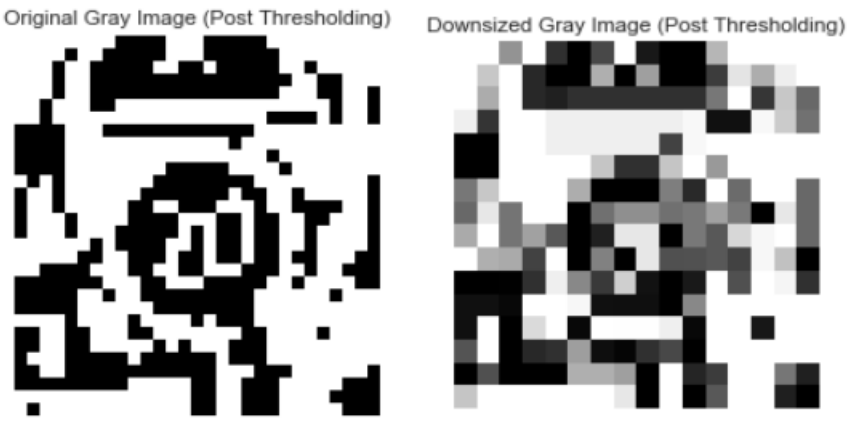


Figure 7: Comparison of thresheld vs. thresheld and downsized images.

All images were downsized to 16x16 afterwards. The downsizing process lost a lot of important information of traffic signs as demonstrated in Figure 7 above. Similar to the first model comparison output using 16x16 images, Gradient Boosting, Multi-layer Perceptron, and XGBoost fit on thresheld images had great cross-validation accuracies of over 80% demonstrated in Figure 8. Gradient Boosting and XGBoost had higher CV accuracies compared to the first output in Figure 5, whereas Multi-layer Perceptron had almost an 8.5% decrease. SVM classifier performed well here, with the cv accuracy of 79.4%. As expected from the previous output, Gradient Boosting and XGBoost required extremely long CV runtime. It was surprising that Gradient Boosting, SVM, and XGBoost had better performance after the thresholding feature extraction, since the loss of information was visually obvious in Figure 7. Although there were other models with better performance, only the SVM and MLP classifiers were chosen as significant due to the long CV runtime.

```
Comparing Different Classifiers:
------------------------------------------------------------------
Random Forest Classifier Acc: 0.693 (+/- 0.02)
CV Runtime: 16.246086359024048
------------------------------------------------------------------
Gradient Boosting Classifier Acc: 0.871 (+/- 0.02)
CV Runtime: 7262.861037969589
------------------------------------------------------------------
Ada Boost Classifier Acc: 0.326 (+/- 0.12)
CV Runtime: 76.97814774513245
------------------------------------------------------------------
SVM Classifier Acc: 0.794 (+/- 0.03)
CV Runtime: 442.92710638046265
------------------------------------------------------------------
Multi-layer Perceptron Classifier Acc: 0.801 (+/- 0.02)
CV Runtime: 515.3083746433258
------------------------------------------------------------------
XGBoost Classifier Acc: 0.858 (+/- 0.02)
CV Runtime: 3526.795161485672
------------------------------------------------------------------
The classifier (with default parameters) with the highest accuracy is
Gradient Boosting Classifier
The cross-validation accuracy is 0.871 (+/- 0.02)
```

Figure 8: Exploratory model comparison output using 16x16 thresheld traffic signs.

As another feature extraction approach, Canny edge detection was performed on the data with the same goal as the thresholding process. In Figure 9, an example of an edge detected image is portrayed. It

outlines the important traffic sign image very well. However, the downsizing ultimately lost a lot of important details, as illustrated in Figure 9 and 10.

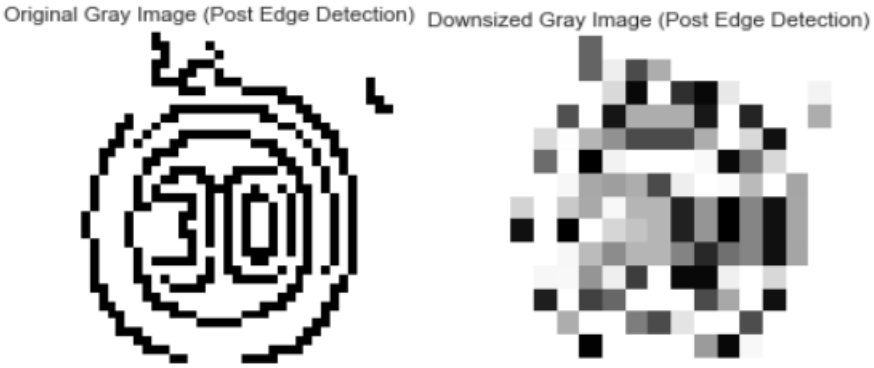


Figure 9: Comparison of edge detected vs. edge detected and downsized images.

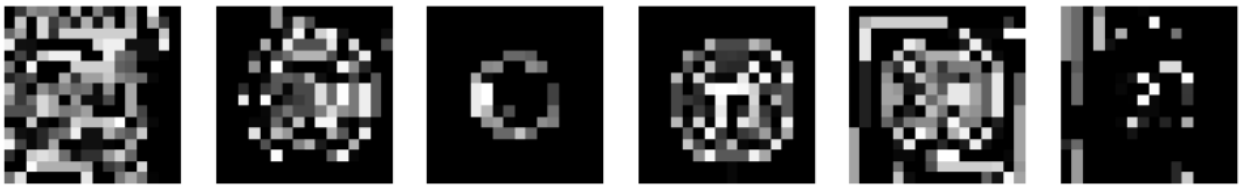


Figure 10: Preview of randomly chosen edge detected and downsized images.

The model comparison output in Figure 11 using canny edge detected images portrays that this feature extraction method did not result in increased CV performances compared to the previous outputs in Figure 5 and 8. The classifier with the highest accuracy was XGBoost, which had a CV accuracy of 54.4%. The downsizing had a negative impact here, which resulted in non-significant performances. Therefore, none of these models were chosen as important.

```
Comparing Different Classifiers:
------------------------------------------------------------------
Random Forest Classifier Acc: 0.362 (+/- 0.02)
CV Runtime: 6.74671196937561
------------------------------------------------------------------
Gradient Boosting Classifier Acc: 0.495 (+/- 0.01)
CV Runtime: 5958.646957397461
------------------------------------------------------------------
Ada Boost Classifier Acc: 0.127 (+/- 0.03)
CV Runtime: 57.12793827056885
------------------------------------------------------------------
SVM Classifier Acc: 0.372 (+/- 0.02)
CV Runtime: 436.39146542549133
------------------------------------------------------------------
Multi-layer Perceptron Classifier Acc: 0.425 (+/- 0.01)
CV Runtime: 329.0484800338745
------------------------------------------------------------------
XGBoost Classifier Acc: 0.544 (+/- 0.03)
CV Runtime: 3275.315999984741
------------------------------------------------------------------
The classifier (with default parameters) with the highest accuracy is
XGBoost Classifier
The cross-validation accuracy is 0.544 (+/- 0.03)
```

Figure 11: Exploratory model comparison output using 16x16 canny edge detected traffic signs.

Principal Component Analysis, a dimension reduction algorithm, was performed on the data. PCA was set up to capture at least 95% of the variance within the input data, with the solver parameter set to whichever solver that returned the minimum number of principal components. Auto, full, arpack and randomized were compared but they all yielded similar results; therefore, auto solver was utilized. The original grayscale data had 256 features (16x16 pixels). Capturing 95% of the variance, PCA was able to reduce the number of features to only 72 principal components. The list containing cumulative % of variance captured and post PCA dimension information is portrayed in Figure 12 below.

```
Solver: auto will be used.
In order to capture at least 95 % of variance within the original data,
72 principal components are needed.

Cumulative % Variance Captured:
 [0.50801126 0.6525222  0.69726492 0.73195126 0.75728661 0.77429116
 0.78736411 0.79764448 0.80775687 0.81672134 0.82463118 0.83121203
 0.83753581 0.84311257 0.84824477 0.85302592 0.85741116 0.86144963
 0.86515702 0.8685259  0.87177512 0.87494432 0.87793641 0.88089271
 0.88376826 0.88654446 0.88924044 0.89169795 0.89405535 0.89638497
 0.89858963 0.90072064 0.90273632 0.90470836 0.90658094 0.90844915
 0.91021231 0.91195981 0.91365323 0.91526358 0.91679509 0.91831409
 0.9198113  0.92124857 0.9226276  0.92400621 0.92535519 0.92666608
 0.92793575 0.92918254 0.93039592 0.93159522 0.93277238 0.93388599
 0.93499084 0.93605802 0.93711834 0.93814392 0.93913805 0.94010691
 0.94106643 0.94198818 0.94288378 0.94375836 0.94462108 0.94545557
 0.94625682 0.9470466  0.94782167 0.94857686 0.94931272 0.95003972]

Shape of the original normalized grayscale data: (8514, 256)
Shape of the newly computed principal components: (8514, 72)
```

Figure 12: Information regarding Principal Component Analysis.

```
Comparing Different Classifiers:
------------------------------------------------------------------
Random Forest Classifier Acc: 0.535 (+/- 0.03)
CV Runtime: 8.21187686920166
------------------------------------------------------------------
Gradient Boosting Classifier Acc: 0.690 (+/- 0.03)
CV Runtime: 3538.1096608638763
------------------------------------------------------------------
Ada Boost Classifier Acc: 0.209 (+/- 0.02)
CV Runtime: 99.93282389640808
------------------------------------------------------------------
SVM Classifier Acc: 0.489 (+/- 0.03)
CV Runtime: 157.5582082271576
------------------------------------------------------------------
Multi-layer Perceptron Classifier Acc: 0.872 (+/- 0.02)
CV Runtime: 171.037828207016
------------------------------------------------------------------
XGBoost Classifier Acc: 0.727 (+/- 0.02)
CV Runtime: 2382.9009857177734
------------------------------------------------------------------
The classifier (with default parameters) with the highest accuracy is
Multi-layer Perceptron Classifier
The cross-validation accuracy is 0.872 (+/- 0.02)
```

Figure 13: Exploratory model comparison output using principal components.

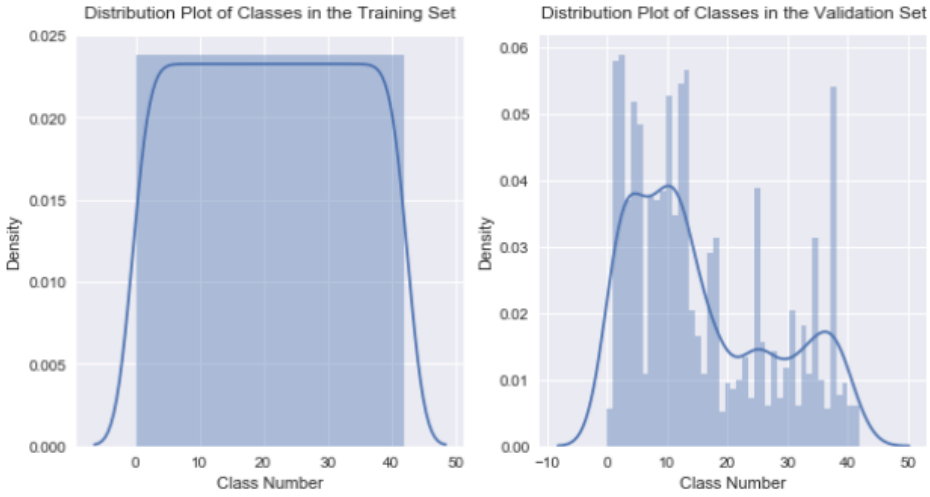
Utilizing principal components led to significant reduction in CV runtime for Gradient Boosting, SVM, Multi-layer Perceptron and XGBoost due to the dimension reduction from PCA. Although the runtime was

a lot shorter, Multi-layer Perceptron classifier demonstrated great CV accuracy of 87.2% as shown in Figure 13. In fact, it performed better than the multi-layer perceptron classifiers with thresheld and edge detected images. Other classifiers had performed better in the previous outputs, so they were not considered as noteworthy.

Wrapper-based feature selection using Random Forest was performed on the image data to select important subset of features. This approach builds many models with different subsets of features using Random Forest and picks the subset that performs the best. As illustrated in Figure 14 below, feature selection chose 64 out of 256 features to be important, which was less number of features than from PCA.

```
Wrapper Select:

Selected: [[71], [72], [85], [86], [87], [88], [89], [90], [91], [100],
[101], [102], [103], [104], [105], [106], [107], [108], [116], [117],
[118], [119], [120], [121], [122], [123], [124], [132], [133], [134],
[135], [136], [137], [138], [139], [140], [148], [149], [150], [151],
[152], [153], [154], [155], [165], [166], [167], [168], [169], [170],
[171], [181], [182], [183], [184], [185], [186], [187], [197], [199],
[200], [202], [203], [216]]
Features (total/selected): 256 / 64
```




```
Total of 8514 images in the training data
Total of 2157 images in the validation data
79.79 percent training set, 20.21 percent validation set
```

Figure 14: Information regarding wrapper-based feature selection (using Random Forest).

In the model comparison output in Figure 15, using feature selection resulted in faster CV runtimes and better accuracies for Random Forest, Gradient Boosting and XGBoost. Random Forest specifically had the fastest runtime and the highest CV accuracy of 80.2% among all the other Random Forest models explored in this project. In addition, MLP classifier had the highest CV accuracy of 89.9% among all the other MLP models in this project. Other models were not remarkable as there were other better performing models from the previous outputs.

```
Comparing Different Classifiers:
------------------------------------------------------------------
Random Forest Classifier Acc: 0.802 (+/- 0.01)
CV Runtime: 5.6491405963897705
------------------------------------------------------------------
Gradient Boosting Classifier Acc: 0.830 (+/- 0.01)
CV Runtime: 2010.6890652179718
------------------------------------------------------------------
Ada Boost Classifier Acc: 0.174 (+/- 0.03)
CV Runtime: 53.2347526550293
------------------------------------------------------------------
SVM Classifier Acc: 0.479 (+/- 0.03)
CV Runtime: 161.04362154006958
------------------------------------------------------------------
Multi-layer Perceptron Classifier Acc: 0.899 (+/- 0.01)
CV Runtime: 235.15239930152893
------------------------------------------------------------------
XGBoost Classifier Acc: 0.832 (+/- 0.01)
CV Runtime: 1018.0708589553833
------------------------------------------------------------------
The classifier (with default parameters) with the highest accuracy is
Multi-layer Perceptron Classifier
The cross-validation accuracy is 0.899 (+/- 0.01)
```

Figure 15: Exploratory model comparison output using feature selection.

Among all feature extraction, dimensionality reduction and feature selection methodologies, wrapper-based feature selection showed significant increase in CV accuracy and decrease in runtime for Random Forest and Multi-layer Perceptron classifiers. Multi-layer Perceptron using principal components had faster CV runtime compared to the MLP using feature selection, but it had a 2.7% decrease in CV accuracy. There were other models that performed well as well. However, since this project is focused on obtaining high performance with limited computing power and time, both the runtime and the cv accuracy need to be compared to evaluate pros and cons of each model. Due to this focus, the top three models selected from the exploratory model fitting phase was the Random Forest using feature selection, MLP using feature selection and MLP using principal components. Surprisingly, these models with high cv accuracies did not require a lot of runtime.

In order to increase model performances, randomizedsearchCV was utilized in the hyper-tuning step. It incorporates the random walk search approach in the given parameter set space. For Random Forest, the parameter grid was set to {"n_estimators": sp_randint(10, 1000), "max_depth": [3, 4, None], "max_features": ['auto', 'log2', None], "min_samples_split": sp_randint(2, 16), "criterion": ["gini", "entropy"]}. For MLP, the parameter grid was set to {'hidden_layer_sizes': sp_randint(10, 500), 'activation': ['logistic', 'tanh', 'relu'], 'solver': ['lbfgs', 'sgd'], 'learning_rate' : ['constant', 'adaptive'], 'max_iter': sp_randint(20, 2000)}. The sp_randint used in the parameter grid above is a scipy function that returns random integers in the input range. RandomizedsearchCV uses this sp_randint function to perform random walk on randomly chosen combination of parameters. It then compares the first parameter to a slightly altered parameter set, in terms of performance scores. If the new parameter set performs better, it continues in that direction in the parameter set space. If not, it goes the opposite way in the parameter set space. Since the learning phase was on balanced data, the scoring metric was set to accuracy. The detailed results of hyper-tuned parameters and the training CV performances for the top three models are shown in Figure 16, 17 and 18.

```
CV Runtime: 12122.635693788528
The best RandomizedSearchCV score: 0.9068592905802209
The best estimator:
 RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=4,
            min_weight_fraction_leaf=0.0, n_estimators=985, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

Figure 16: Hyper-tuned parameters for the Random Forest model using feature selection.

```
CV Runtime: 25216.135255098343
The best RandomizedSearchCV score: 0.9248296922715528
The best estimator:
 MLPClassifier(activation='tanh', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=499, learning_rate='constant',
       learning_rate_init=0.001, max_iter=600, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=None, shuffle=True, solver='lbfgs', tol=0.0001,
       validation_fraction=0.1, verbose=False, warm_start=False)
```

Figure 17: Hyper-tuned parameters for the MLP model using feature selection.

```
CV Runtime: 12382.06505227089
The best RandomizedSearchCV score: 0.8726802912849424
The best estimator:
 MLPClassifier(activation='tanh', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=394, learning_rate='constant',
       learning_rate_init=0.001, max_iter=1690, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=None, shuffle=True, solver='lbfgs', tol=0.0001,
       validation_fraction=0.1, verbose=False, warm_start=False)
```

Figure 18: Hyper-tuned parameters for the MLP model using principal components.

Surprisingly, the hyper-tuned Random Forest model using feature selection had higher training CV accuracy (of 90.7%) than the MLP model using principal components. The MLP model using feature selection had the highest CV accuracy of 92.5% among all. The CV runtime for the hyper-tuning process of Random Forest took just slightly shorter than the MLP using principal components. This is because of the high n_estimator, which represents the number of trees in the forest. Since there are a big number of trees that the input image data must go through, it ultimately led to longer runtime. Regardless, it was the fastest among top three models. The MLP models also had large numbers of hidden layer nodes, the MLP model using feature selection having the biggest (499 nodes). This model had the longest CV runtime for the hyper-tuning process due to the large hidden layer nodes.

Results and Discussions

The models with hyper-tuned parameters were tested on the unbalanced validation dataset to obtain the true model performance on the data with the original integrity. Due to the imbalance in the validation set, the weighted average of precision, recall and f-1 scores rather than accuracy were utilized as the performance metrics. Testing accuracy is not a good metric to use, when there are extreme class imbalance issues. High precision and recall scores illustrate that the classifier is returning accurate results (precision), as well as majority of all positive results (recall). An ideal algorithm with high precision and recall will

return many predictions classified correctly. As expected from the training CV accuracy, the MLP model using feature selection had the highest testing precision (0.92), recall (0.91) and f-score (0.91) among all, portrayed in Table 1 below. The Random Forest model using feature selection came in second in terms of precision (0.89), recall (0.88) and f-score (0.88). Although the MLP model using principal components had promising training CV accuracy in Figure 13 before the hyper-tuning phase, the testing precision, recall and f-score all hovered around 0.82~0.83. As shown in the confusion matrices in Figure 19, 20 and 21, this model was unable to correctly classify some of the highly populated class compared to the other two models, which resulted in lower performance scores.

Classifiers	Training (cv = 5) Accuracy	Testing Accuracy	Precision Score	Recall Score	F-Score
Feature Selected (Wrapper-based) Random Forest Classifier	0.905826	0.878999	0.887589	0.878999	0.879842
Feature Selected (Wrapper-based) Multi-layer Perceptron Classifier	0.924114	0.910524	0.916128	0.910524	0.911308
Multi-layer Perceptron Classifier w/ Principal Components	0.866544	0.823829	0.833671	0.823829	0.825159

Table 1: Performance result table containing final 3 models.

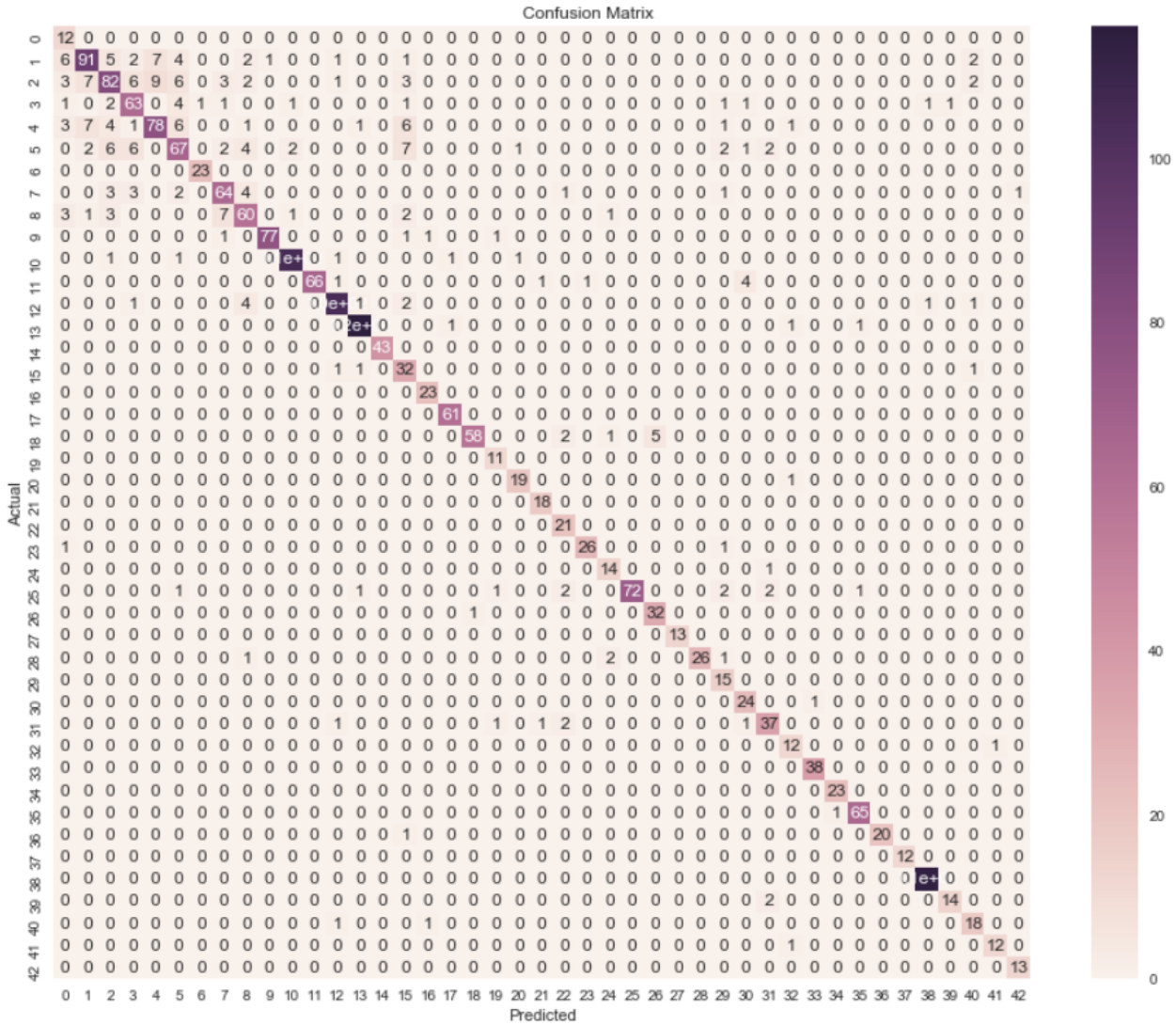


Figure 19: Confusion matrix of hyper-tuned Random Forest model using feature selection.

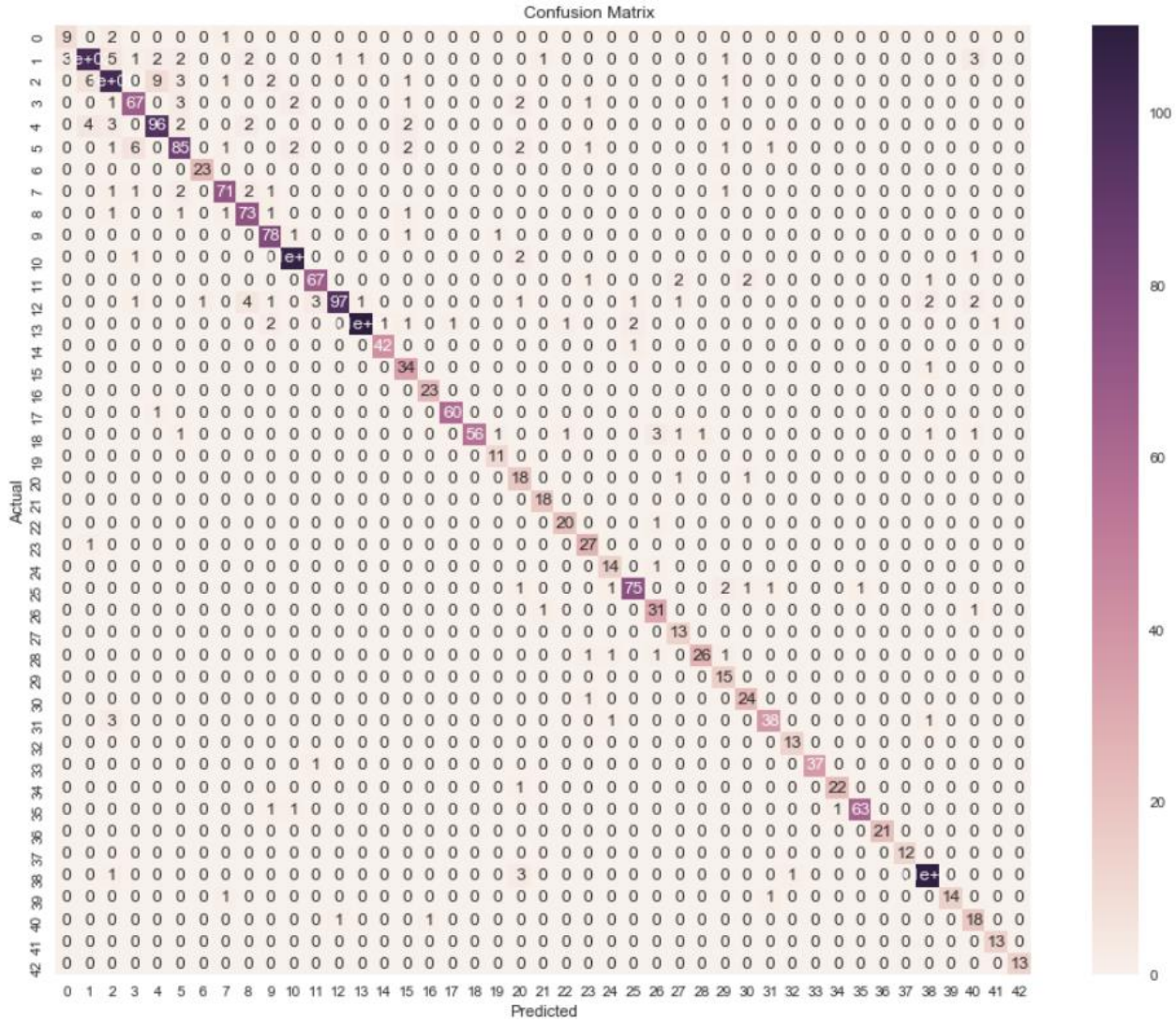


Figure 20: Confusion matrix of hyper-tuned MLP model using feature selection.

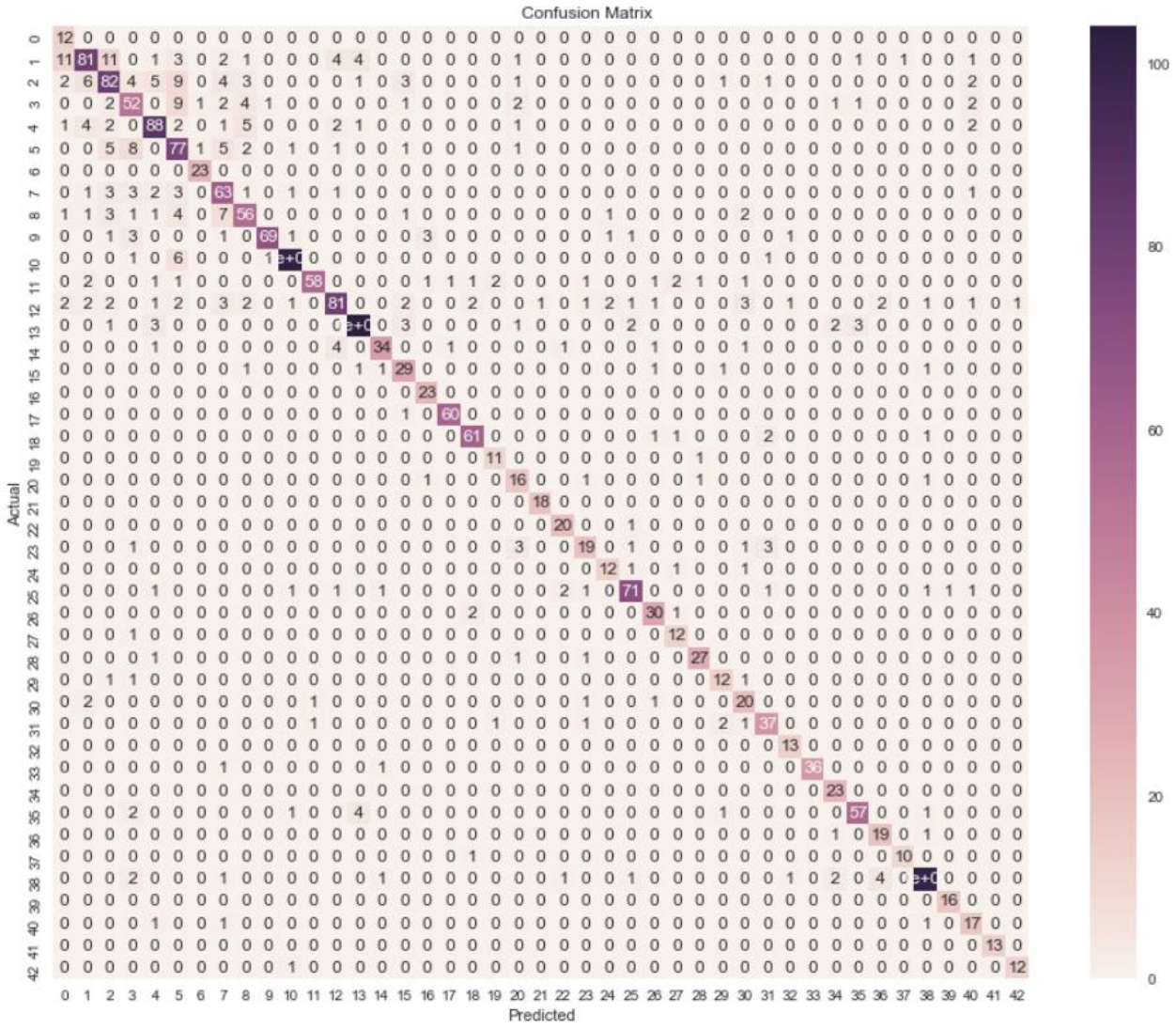


Figure 21: Confusion matrix of hyper-tuned MLP model using principal components.

Considering the performance, time and computing power taken, only the Random Forest and MLP model using feature selection were chosen as the final models. The wrapper-based feature selection methods performed the best, because it essentially only selected the feature subsets that contained the object of interest. Since the images were non-centered, it selected bigger range of feature subsets than it would select on perfectly centered data. This ultimately led to reducing time and computing costs, in addition to the improved performances.

Conclusions and Future Work

Effectiveness of many classifiers were analyzed, fit on low quality (both in dimension size and color) and non-centered traffic sign image data in this project. Although the learning phase was restricted in terms of information due to the dimension downsizing, Random Forest and Multi-layer Perceptron classifiers with hyper-tuned parameters still illustrated significant performances. The feature extraction methods (thresholding and canny edge detection) were able to remove a lot of unimportant clutter and capture the traffic signs within each image before the size downsizing. However, they were all affected heavily by the downsizing process and hence resulted in low performances. Dimensionality reduction (PCA)

and feature selection (wrapper-based using Random Forest) both showed sizeable decreases in runtimes and increase in model performances. The Figure 22 below shows an example of the exploratory model comparisons step using a 32x32 grayscale images. The performances were not drastically different; however, the CV runtime was extremely longer due to the bigger dimension size. Although utilizing the 32x32 images could have led to better performing models, it is proven in this project that lower quality data can also bring significant results with faster runtimes.

```
Comparing Different Classifiers:
------------------------------------------------------------------
Random Forest Classifier Acc: 0.724 (+/- 0.04)
CV Runtime: 24.647209405899048
------------------------------------------------------------------
Gradient Boosting Classifier Acc: 0.810 (+/- 0.04)
CV Runtime: 19981.489895820618
------------------------------------------------------------------
Ada Boost Classifier Acc: 0.116 (+/- 0.05)
CV Runtime: 286.26625180244446
------------------------------------------------------------------
Multi-layer Perceptron Classifier Acc: 0.877 (+/- 0.02)
CV Runtime: 625.529922246933
------------------------------------------------------------------
XGBoost Classifier Acc: 0.839 (+/- 0.03)
CV Runtime: 13988.283029556274
------------------------------------------------------------------
The classifier (with default parameters) with the highest accuracy is Multi-layer Perceptron Classifier
The cross-validation accuracy is 0.877 (+/- 0.02)
```

Figure 22: Exploratory model comparison output using 32x32 grayscale traffic signs.

In order to further develop this multi-class classification project, a combination of feature extraction and dimension reduction/feature selection approaches could be explored to increase model performances. Since the feature extraction processes were heavily affected by downsizing as shown in Figure 7 and 9, implementing dimensionality reduction or feature selection algorithms on the feature extracted images without the downsizing phase to save time and computing costs will be interesting to explore. In addition, exploring with different randomizedsearchCV parameter grid will be beneficial. For example, the Multi-layer Perceptron model is proven to perform well on image data. Increasing the number of hidden layers will make the model perform significantly better as proposed in the references; however, it was not explored in this project due to the extremely limited time and computing power. Another way of developing this project is by providing some bias to minority classes by using compute_class_weight and class_weight parameter while training the model on the unbalanced data (without using under-sampling). This will provide some bias towards the minority classes while training the model, which will ultimately help improving performance of the model; however, it will also require more computing power because of the increased instances in the learning phase from the populated class, compared to the under-sampled data.

Computer vision is utilized in many fields that want to receive beneficial assistances from technology. Although technology is improving every day, there are still limitations in making devices (IoT devices, for example) portable and powerful enough to perform necessary tasks in a timely manner. Some may utilize over-the-air services to achieve fast computing power, but there are times where local computation is necessary. Being able to use not-so-perfect data and converting them into useful information can bring tremendous advantage over other competitors. Further research and expanding this project will bring a big impact in the computer vision market.

References

[1] Baro, Xavier, Sergio Escalera, Jordi Vitria, Oriol Pujol and Petia Radeva, "Traffic Sign Recognition using Evolutionary Adaboost detection and Forest-ECOC classification", http://www.maia.ub.es/~sergio/files/Transport09.pdf

[2] Greenhalgh, Jack and Majid Mirmehdi, "TRAFFIC SIGN RECOGNITION USING MSER AND RANDOM FORESTS", http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.304.5878&rep=rep1&type=pdf

[3] Sermanet, Pierre and Yann LeCun, "Traffic Sign Recognition with Multi-Scale Convolutional Networks", http://yann.lecun.com/exdb/publis/pdf/sermanet-ijcnn-11.pdf

[4] Stallkamp, Johannes, Marc Schlipsing and Jan Salmen, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition", https://www.ini.rub.de/upload/file/1470692848_f03494010c16c36bab9e/StallkampEtAl_GTSRB_IJCNN2011.pdf

[5] Yang, Yuchen, Shuo Liu, Wei Ma, Qiuyuan Wang and Zheng Liu, "Efficient Traffic-Sign Recognition with Scale-aware CNN", https://arxiv.org/pdf/1805.12289.pdf