Project One: Phase One Report

Vinayak Joshi
Department of Engineering (Computer Systems)
The University of Auckland
Auckland, New Zealand

***Abstract* — Upon the setup of two different machine learning models, the multilayered perceptron (MLP) and support vector machine (SVM) to classify various traffic signs, we find that the MLP model tends to perform better in terms of accuracy, precision, and recall after some simple cross validation of a dataset of over 73,000 points. The choice on which to use, however, is still dependent on context.**

I. Introduction

The following report details the training of two machine learning models: SVM (Support Vector Machine) and MLP (Multilayer Perceptron). These two models were fed a large dataset consisting of images of exactly 43 different traffic signs for the purposes of classification. In the forthcoming sections, we will delve into the preprocessing of the data, model designs and configurations, experimental setups, and a final comparative analysis of the two models based on various performance metrics. Attached with this report are two python files which underpin the flow of the reports exploratory nature to set up, validate, and test each model which you may use to supplement the following reading.

II. Melodies of the Machines

A. Data-sets and applications

Mentioned in the introduction, the images used for classification comprise of 43 different traffic signs, including speed limits, roadworks, hazards, crossings, vehicle constraints, and more. The application of such classification includes but is definitely not limited to things such as autonomous vehicle software, traffic flow analysis, navigational software, traffic management, data collection, infrastructure planning, etc.

B. Data Preprocessing

Both machine learning models were fed the exact same data — a two dimensional array whose elements are one dimensional arrays containing the pixel values for an image within the data-set. The sub-arrays are of length 3072 as each image is 32x32 pixels with standard RGB color depth, or 32x32x3. This two dimensional array was created from a flattened fourth dimensional array, where each element consisted of the original 32x32x3 raw pixel data, thus describing our first preprocessing step. The justification for this is the simple fact that both SVM and MLP demand inputs as feature vectors, which must be supplied as a single dimensional array.

The second preprocessing step was splitting the image data into two categories: testing, training, and validation the obvious reason being that using training data for testing means our predictions would be 100% accurate as the model has already seen those particular data points. If we allowed this, we would fail to make any accurate assessments of the models on unseen data, with the same principles applying to tweaking our model using the testing data instead of the validation data. Our third preprocessing step is label encoding. Both MLP and SVM models are supervised, and thus we require the data to be labeled. A separate label array was created alongside the image arrays which kept track of what class each element belonged to as an integer, in our case a range from zero to 42.

This leads to our fourth preprocessing step, which involved shuffling the data prior to splitting. While this isn't strictly necessary, it means that we avoid the risk of taking into account the order of the images, which is completely irrelevant to the classification of traffic signs.

Let us first now perform some cross validation on the MLP model. For the hyper-parameters, 10,000 iterations with a step size of 0.001 was tested to perform very well, with an 80 - 10 - 10 training, validation and testing split. Shown below is the confusion matrix for this experimental setup. scaled to a range from zero to one, as the sample size of images for each class of traffic sign varies quite a bit ranging from 390 all the way up to 4,920 images.

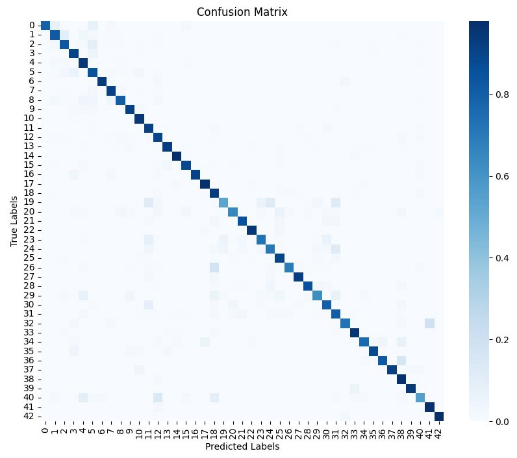


fig 1 – confusion matrix for an MLP model with no pixel value preprocessing

The model appears okay with some inaccuracies, and there are some classes which appear to suffer from poor precision, as seen by the vertical streaks scattered throughout the predicted axis. There are some preprocessing steps we can take by inspecting the data.

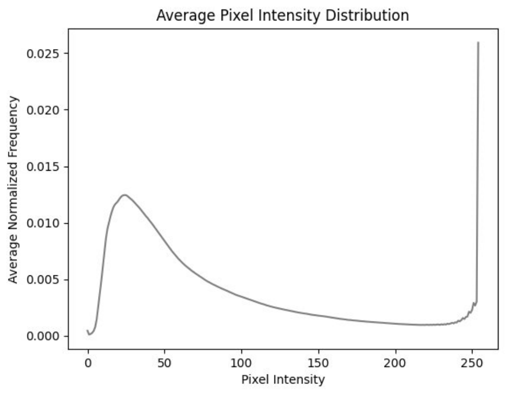
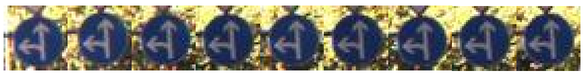


fig 2 — pixel value distribution across all images

Taking a look at figure two, we can see that there is a huge spike in pixel values greater than ~240. Inspecting this on a per-class basis reveals much the same thing; that in many images, the lighter colors far outweigh the darker ones. Inspecting our data visually, we begin to see why this is the case, and how we can best tackle it.



Above is a series of nine different images from class number 37. Notice how the background is dominated with bright pixels from the daylight environment. Inspecting all other classes we see this is actually a common theme — many of the photos taken are in bright daytime conditions, with a few being taken in low-light conditions. There are also many instances of signs that captured the extreme glare of the sun such as the one shown below, further contributing to the high number of extremely bright pixels.

In fact, the pictures of signs taken in their closest proximity make up no more than ~50% of the image area on average through an informal analysis, where it was found for round traffic signs, the distance from the edges closest to the border of the image was ~3 pixels at the minimum, and so we can compute the area of such traffic signs as:

$$area_{sign} \approx \pi r^2, where\ r = \left(\frac{width_{img} - 3 \cdot 2}{2}\right) = \frac{32-6}{2} = 13$$

$$area_{sign} \approx \pi \cdot 13^2 \approx 530.92$$

Which represents around 50% of the total image area. For triangular traffic signs, many of them fit within the circular region, and would comprise *even less* of the image area — these two types of signs making up the vast majority of our data. This means that for many samples, we have a large amount of noise that we should look to consider.

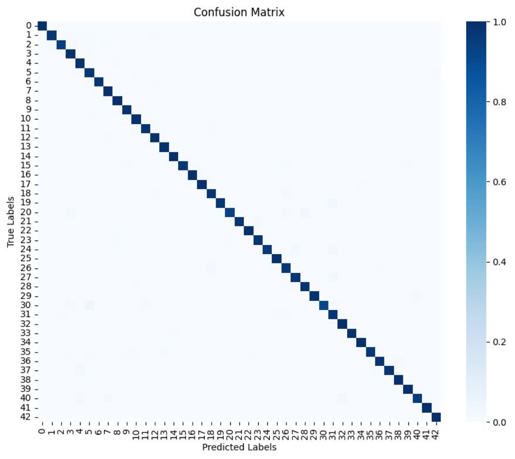


fig 3 — confusion matrix for MLP after 'robust' scaling

As evident from our graph in figure two, this could skew our results as the brighter pixels are 'heavy' on the data. A great way to minimize the effect of such dominant data, given that we know this is coming from a feature completely unrelated to traffic signs, is to treat these as outliers through the use of a robust scaler — which transforms data points by subtracting the median and dividing by the inter-quartile range, making the data more robust in the face of outliers. In figure three, we see our new results using the "RobustScaler" function from sci-kit learn's library.

It is very clear that the confusion matrix is far, far better. There are still a few inaccuracies, but they are nearly invisible on the graph and our precision errors have almost completely disappeared suggesting that our model is in a very good state. Various other scalers were also used, such as the "MinMaxScaler" and "StandardScaler", with the "RobustScaler" producing the best results for the aforementioned reasons.

We will now move onto our SVM model, and perform much of the same validation steps before moving onto testing. First, let us see the model performance with the standard hyper-parameters; a *C* value of one with the kernel set to the radial basis function. For the same reasons mentioned prior, we will also flatten the images, split them into testing and training sets, encode the labels, and shuffle everything prior to input. Below is a confusion matrix of the result.

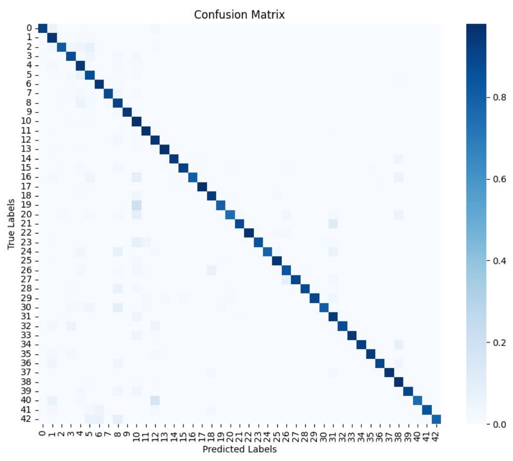


fig 4 — confusion matrix for an SVM classifier with no pixel value preprocessing

Overall, accuracy is okay, but we suffer once more from precision errors from predicted classes particularly those in the upper and lower ranges. We know for a fact that in SVM the decision boundary is determined by the support vectors alone, and hence the justification we used to scale the data in the MLP classifier won't work well for the SVM classifier. Let's first tackle the precision issue by inspecting classes eight, ten, 31 and 38 which seem to have the worst precision.

Mentioned prior was the fact that the data-set is quite imbalanced. In fact, if we look at the classes that are giving us the most trouble in our SVM model, they are among the ones that contain the most data. Class eight contains 2,670 images, ten contains 3,810 items, 31 contains 1,470 and 38 a whopping 3,930. The implication of this is quite obvious, our imbalanced dataset means that we are more biased towards these particular traffic lights — hence why the model predicts these particular classes more often than it should.

We can deal with this by oversampling our data. While we could opt for undersampling, this would mean cutting potentially up to 3,500+ images from the largest sample sets, where we could lose a lot of valuable data in the process.

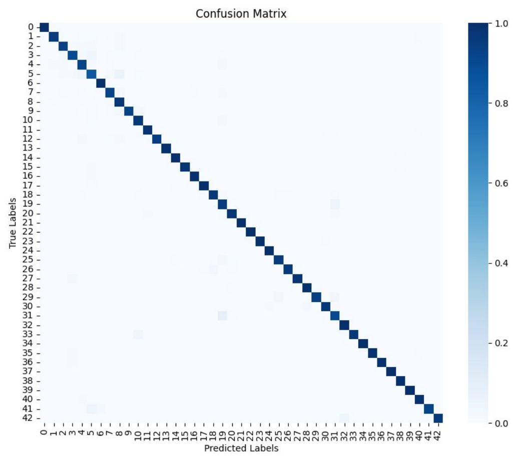


fig 4 — SVM model predictions after oversampling minority classes

We can see that our model performs far, far better. Accuracy, precision, and specificity have all improved. Most of the inaccuracies have fallen wayside, with only a few pale blue squares visible.

III. The Great Filter

Let us now finally test our refined models with the remaining ten percent of the data which the model has not yet seen. We will compute the F-1 score, accuracy, precision, and recall.

Below is a table comparing the two models' performances. Given that there will be 43 different instances of the aforementioned metrics, we will simply consider the mean.

COMPARATIVE ANALYSIS	MLP	SVM
Accuracy	0.9840	0.9533
Avg. Precision	0.9841	0.9582
Avg. Recall	0.9840	0.9533
Avg. F1 score	0.9839	0.9547

It is very clear that the performance difference between the two models is somewhat small in terms of precision, recall, and f1 scores.

As it stands, the MLP classifier is a little more accurate and precise on average in addition to a higher overall recall and f1 score. In the context of traffic signs, there are many avenues we could go down in terms of prioritization depending on our application. For example, if our goal is to minimize loss of human life whilst driving an autonomous vehicle, it may be better to sacrifice accuracy in classifying signs that indicate horseback riding is not allowed in favor of, say, stop signs.

For now with no context on the application of both models, neither can really be chosen over the other. For general use, it may be reasonable simply go with MLP as it performs better in nearly all the metrics shown in the table. The general procedure to determine which would be better to use for any given scenario is to create a priority list which contains traffic signs that one cannot afford to misclassify, check both models for specific accuracy, precision, recall and f1-score for each one, and tune the models accordingly. It would also be useful to analyze both models' ROC curves before deployment for real world use. Recall that SVM models are typically faster at classifying, which may give it an edge over the MLP model in time-sensitive scenarios.

Please note that the data used to train the MLP model was precisely the same data used to train the SVM model, as well the validation data for tweaking and final testing data. This was done by initially splitting the data into an 80-20 partition, then splitting the 20 partition into 10-10 partitions through sklearns "train_test_split" function, and saving each split as ".joblib" files, where these were then used.