COMPSYS 306 – Assignment Pt. 1 Report

Jake Bowden (jbow177)
Software Engineering – Part 3
University of Auckland
Auckland, New Zealand

I. Dataset and Application Description

A. Dataset

This dataset of traffic signs contains 73,139 total images of traffic signs, split across 43 distinct categories or types of traffic signs. Each image is 32 by 32 pixels in size, in full colour. The smallest of the categories contains ~450 images, while the largest contains ~4900 images. Also, the images in the dataset seem to have quite a large variation in light and contrast. For example, some images are very "well lit", while others are mostly dark, and others have a dark background with a noticeably light traffic sign. However, these are all important to keep in the dataset to ensure that we can classify under a large variety of lighting conditions.

B. Application

My goal is to train a multi-layer perceptron (MLP) model, as well as a support vector machine (SVM) model on this data, to attempt to classify it. Then, I will compare the results of the two models.

II. Data Pre-Processing

A. Normalization

As my data does have quite a large value range, I normalized it to alleviate this (since the dataset would not be likely to follow a standard distribution), which ended up being an implicit step in ensuring that all images are the same size. However, as Histogram of Oriented Gradients (HOG) later became a step in my pre-processing, this turned out to not be too necessary, as the image pixels were no longer machine-learning inputs.

B. Feature Extraction - HOG

For the MLP model, I was originally going to purely use the model to extract the features I need organically, but that led to very inaccurate results. Therefore, for both models I used HOG to find features for each image. I was also considering using SIFT (Scale Invariant Feature Transform) as I thought that since there is no variation within the same sign usually, it would be particularly good with the way it finds common parts of an image, since the background pixels can easily contribute to overfitting. However, it did seem to be more computationally intensive than HOG, so I tried to only use HOG, which gave me a good enough result, so I kept that as my only extracted feature.

C. Data Split

I split my data into three parts – for training, validation, and testing. This was done with two splits, with the first one created being the testing dataset, which was 10% of the total dataset. The second split created was the validation dataset, which was 20% of the remaining dataset, with the training dataset being the remaining 80%. This 3-way split was done so that after I tune my hyper-parameters, I can make sure that my tuning was not just overfitting the validation dataset, by using the testing dataset.

Also, I made a larger chunk of my dataset training (20% validation split instead of 30%, and low testing split) as the dataset is exceptionally large for each category, meaning that there still should be plenty of data to validate / test my model on.

III. Model Design and Configurations

Note that most information about the MLP and SVM models is described in the next section. Other than that, they are both models taken from Scikit-Learn, with MLP being from MLPClassifier, and SVM being from svm.

IV. Experimental Setup

A. Hyper-parameter tuning - MLP

For the hyper-parameters, I started at an initial learning rate of 0.3, and a maximum iteration amount of 120, as a sort of low estimate of the number of iterations that I would want to perform (as I wouldn't want to go above a learning rate of 0.4 to avoid it becoming unstable, or below 100 iterations as this would be too short for my scenario, possibly causing underfitting). This was not a very good model, with an accuracy score of ~50%. Therefore, I slowly increased the maximum iteration count and then decreased the initial learning rate, also setting the learning rate to be adaptive, to make sure that convergence occurs. I also added in early stopping, as with having many epochs, my model could overfit the data. With this iterative process, my accuracy score increased to ~89% and I was not able to get it higher. The hyper-parameters that gave me this result were an initial learning rate of 0.01 and 2500 max iterations.

The hidden layers could also be considered hyper-parameters, being sizes of 144 and 72 (as my input layer had 288 neurons and my output layer was a size of 43). Changing the number of layers and the size of them did not significantly impact the results (such as adding a third layer or removing the second layer, or changing the neurons in each layer), so I kept them as is. The number of layers was kept low to avoid computational costs and vanishing gradient.

B. Hyper-parameter tuning - SVM

Due to my model having exceptionally good results on the first test, I decided to not change my hyper-parameters from their initial values. These hyper-parameters included using a kernel of radial basis function (RBF), gamma of 0.7, C of 3 and 1400 for the number of maximum iterations. While this did not reach convergence, the results were particularly satisfactory, so I decided to not increase the max iterations (or any other parameter). Note that these hyper-parameters were initially chosen as they appeared to be quite standard.

Also, due to my suspicion in my validation and testing results being inaccurate, I created a function to test random pictures, showing the expected and the actual picture category number. As these mostly lined up I gained confidence in my results.

C. HOG Setup

For my use of HOG, I did a bit of visual experimentation before the machine learning part to make sure that the HOG would effectively visualize the important image details. Therefore, I ended up with 8x8 pixels per cell, and 2x2 cells per block (meaning 2x2 blocks per image), along with 8 orientations per cell. This was not too computationally expensive at all, despite having to unflatten the image data before (flattening the image in the first place was unnecessary for machine learning in hindsight).

This setup meant that the outlines of the sign could be shown effectively, without including too much unnecessary data on the background behind the sign. It also allowed the sign imagery / text to be somewhat visible, as it uses a lot of lines there anyway. The contents of the sign are also more accessible by the model, as the blocks being 16x16 pixels means that the center block (that commonly contains most of the contents) can be easily used to significantly help to classify the traffic signs.

V. Comparative analysis

Here I will show the tables of my model results (note that for Precision, Recall and F1-score, these are taken as “macro” as “micro” does not show a difference between scores (for the Scikit-Learn Metrics functions)).

A. Table 1

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
MLP	88.54%	89.26%	88.90%	88.88%
SVM	97.03%	98.11%	97.30%	97.69%

Fig. 1. Figure of evaluation table of the two models

From this table, it appears that both models have respectable results, with all values being over 85%. These results are further discussed below:

- The high accuracy suggests that for most predictions concerning a category of signs, the prediction is correct that it is or is not that sign most of the time. As all signs are important to predict correctly to be safe and stay lawful, this is very important to be high.
- The high precision suggests that for when a category is guessed, most of the time it is actually that category, instead of another. This is probably most important for signs of slightly lower importance like “bumpy road”, as confusing that for “one-way” signs could be very bad.
- The high recall suggests that for road signs, we will mostly guess them if it is one, instead of thinking it is another one, which is really important for speed limit signs, as we wouldn’t want to guess the wrong speed limit.
- The high F1-score also suggests good accuracy, as it takes into account recall and precision. Therefore, this would be good to be high for the same reasons that it is good that accuracy is high.

It is worth noting that there was some variation in these scores for each category, when I looked at the Scikit-Learn Metrics classification report for MLP, with some categories having a recall of below 70%, with the precision and F1-score also being a bit lower than their usual values. This likely suggests that those categories are guessed less often than they should be, while others are guessed in place of them. This includes the speed limit sign, the dangerous curves sign, and the pedestrian sign. The most concerning out of these is probably the dangerous curves sign being under-guessed, as not being seen could cause dangerous accidents.

Overall, it seems that SVM had the better results, with higher metrics across the board.