Machine Learning Foundation Training (Batch 04)

Capstone Project
Sign Classification

Dilshan Gunarathna

DSA_0388

Table of Contents

Table of Figures

1. Introduction

Image processing with machine learning architectures is a popular area of interest in both research and technical applications. A major application of such image processing architectures is in sign classification. In sign classification, it is important to have an accurate system as well as the ability to quickly response. Emerging machine learning architectures are capable of providing both these aspects. With the developing technology, the computational complexity and hardware requirements for machine learning are also reducing by each day.

Simple image processing techniques are rapidly being replaced by new machine learning algorithms. One reason for this is easy and efficient identification with machine learning algorithm. These algorithms are capable of recognizing patterns with much less time and effort. Another reason is that machine learning algorithms are easier to automate. With minimum number of commands, they are able to self-improve. Moreover, these algorithms can process multi-dimensional data and comprehend complex algorithms that are difficult to operate manually.

Another area that draws attention regarding to image recognition is deep learning architectures. This is a subgroup of machine learning. However, deep learning is more advanced than traditional machine learning algorithms. They can process very large amounts of data with much less time than traditional machine learning methods. Also, they need less human intervention. For traditional machine learning, the input data should be processed, and certain features need to be defined before sending into the model. But deep learning algorithms are capable of learning those features on their own.

In this study, we are proposing a traditional machine learning algorithm with Support Vector Machines (SVM) and a deep learning algorithm with Convolutional Neural Networks (CNN) for traffic sign recognition. We tried to improve both algorithms in terms of accuracy, in order to investigate the best suited method. The algorithms were tested on a created database consisting of traffic sign images.

2. Background

2.1. SVM

SVM is a widely used machine learning algorithm for classification problems. It maps the input data into a higher dimensional feature space using nonlinear mapping. In the feature plane, an optimal hyperplane is constructed to determine the outputs. Even though this algorithm was first introduced for addressing two-class classification problems, later it was extended to use with multi-class classification problems by decomposing a general problem into binary classification problems.

The algorithm is associated with a loss function which should be minimized in order to achieve the optimal results for a given dataset. There are hyperparameters regarding this loss function we can choose for the optimization of the model.

2.2. CNN

CNN is a deep learning algorithm mostly used for image detection and recognition tasks. Due to the precise, yet simplified algorithms, CNN is popular in difficult pattern recognition systems.

Like a biological neural network, a CNN also has neurons in each of its layers. They can self-optimize through the learning process, while performing operations on the input dataset.

CNN algorithms have three types of layers: convolutional, pooling, and fully connected. When these different types of layers are stacked together in a specific manner, a CNN is formed. Each layer performs different tasks. In the convolutional layers, different filters are applied to input images and feature detection is achieved. Pooling layers introduce translation invariance, which helps to detect common features in the inputs such as edges. Finally, the fully connected layers have neurons arranged in the same way as traditional artificial neural networks.

3. Methodology

First, we Download a database consisting of traffic sign images. Considering the amount of data in the created database, we decided to approach the classification problem in both SVM and CNN algorithms and find the best suited algorithm.

3.1. Database Generation

The database consists of images of traffic signs. The images were labelled into following 8 groups:

1. 20 signs
2. 30 signs
3. 50 signs
4. 60 signs
5. 70 signs
6. 80 signs
7. 100 signs
8. 120 signs

The classification problem is designed to identify which of these groups a new image falls into. With this approach, instead of identifying all the different traffic sign images, we were trying to generalize the problem into identifying the sign shown in Fig. 1.

Figure 1: signs

3.2. SVM Approach

The first step toward this approach was to optimize the hyperparameters in the SVM loss function. Python sklearn library has the function GridSearchCV which enables hyperparameter optimization for a given dataset. We used this function to determine the regularization parameter, kernel, and the kernel parameter. The highest accuracy was obtained for the rbf kernel. The optimized parameters were added to the SVM model and accuracy was observed.

3.3. CNN Approach

The CNN algorithm was implemented in Python using the keras library. CNN is a mathematical construction, usually composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two are the convolutional layer and the pooling layer, which perform feature extraction, while the third is a fully connected layer that maps the extracted features to the final output, such as classification. The convolutional layer plays a key role in CNN, and it consists of a bunch of mathematical operations.

We used Categorical Cross-Entropy as the loss function because minimizing the cross entropy is equivalent to maximizing the lower bound of the mutual information between the traces and the secret variable. Optimization is done using the Adam approach on batch size 50.

In CNN, multiple hyperparameters should be carefully selected to retrieve the best classification rate, including model hyperparameters that define the CNN architecture and optimized hyperparameters (such as loss function, learning rate, etc.). The parameter values are usually selected based on literature and a process of trial and error (by running experiments with multiple values). The conventional method is to start with the CNN architecture adopted in the field similar to ours, and then update the hyperparameters through experiments.

4. Results

4.1. SVM Results

For the SVM algorithm with upsampling, Gaussian smoothing filter, and HOG, the obtained accuracy was 99.95%. The confusion matrix for the algorithm including upsampling, Gaussian filter, and HOG is given in Fig 3.

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        52
           1       0.95      0.98      0.97       461
           2       0.95      0.96      0.96       485
           3       0.98      0.94      0.96       308
           4       1.00      0.99      0.99       474
           5       0.95      0.94      0.95       425
           6       0.98      0.97      0.98       334
           7       0.97      0.98      0.98       310

    accuracy                           0.97      2849
   macro avg       0.97      0.97      0.97      2849
weighted avg       0.97      0.97      0.97      2849

Testing  set score for SVM: 0.968410


Training  set score for SVM: 0.999532
```

Figure 2: Accuracy, Precision, Recall and F1-Score

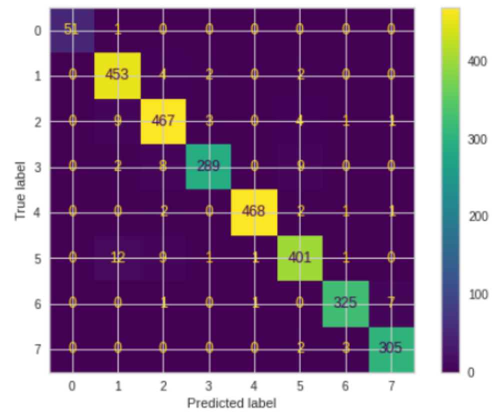

Figure 3: Confusion matrix with pre-processing and upsampling

In the confusion matrices, 0 denotes the 20 sign images, 1 denotes 30 sign images, 2 denotes the 50 sign images, 3 denotes 60 sign images, 4 denotes the 70 sign images, 5 denotes 80 sign images, 6 denotes the 100 sign images and 7 denotes the 120 sign images.

4.2. CNN Results

The accuracy obtained for the model derived from Adam optimizer was 98.63%. The variation of accuracy, validation accuracy, val loss and the loss with epoch for dropout given in Fig. 4.

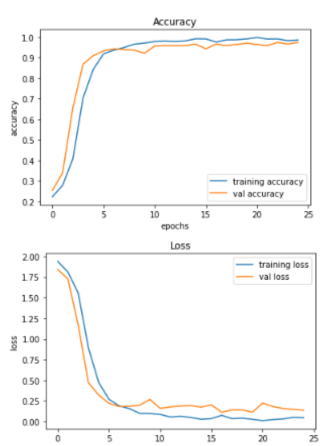

Figure 4: Accuracy and loss

y_pred	0	1	2	3	4	5	6	7	All
y_act									
0	6	0	0	0	0	0	0	1	7
1	0	88	2	0	0	0	0	0	90
2	0	0	84	0	0	0	0	0	84
3	0	0	0	68	0	2	0	1	71
4	0	0	0	0	84	0	0	0	84
5	0	0	0	3	0	54	0	1	58
6	0	0	0	0	0	1	59	0	60
7	0	0	0	0	1	1	0	56	58
All	6	88	86	71	85	58	59	59	512

Figure 5: Confusion matrix

5. Discussion

The SVM model derived from hyperparameter optimization has given an accuracy of 99.95%. However, the processing time was increased for this case since pre-processing and feature extraction takes additional time to process. In Fig. 3, the confusion matrix shows that the model gives a considerable number of wrong predictions regarding classes.

Now, let us consider the results of the CNN model. The accuracy of the CNN model is 98.63%, the validation accuracy is 97.46%. Separate validation and test sets are required because training a model always requires fine-tuning its hyperparameters and model selection. Since this process is performed based on the performance of the validation set, some information about that validation set goes into the model itself, i.e., overfitting to the validation set, although the model for the learnable parameters is never trained directly on it. Because of this, the model is guaranteed to perform well with precisely matched hyperparameters in the validation set for the same validation set. Hence, a completely invisible data set, i.e., a separate test set, required to properly assess model performance, as model performance provides us with data never seen before, i.e., the generalizability.

6. Conclusions

In this study, we have investigated a suitable machine learning algorithm for traffic sign recognition. Two approaches were taken into consideration: SVM algorithm from traditional machine learning architectures and CNN from deep learning architectures.

A database consists of traffic sign images. Then, according to this database, the SVM and CNN models were trained to achieve the best possible test accuracy. In the SVM model, the achieved accuracy was 99.95%. The accuracy of the CNN model was 98.63% and the validation accuracy was 97.46%.