# Machine Learning - Programming Assignment
# Road Signs Classification

Fabio Zamboni

06-07-2023

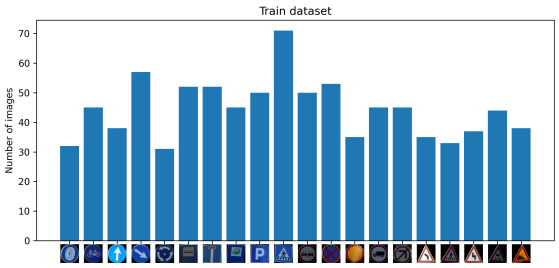
## 1 Introduction

Autonomous self driving vehicles are the future of mobility and an important part of autonomous driving is sign recognition. Specifically in this paper we are going to study a case of vertical sign recognition. The dataset provided contains 20 different road signs, with a total of 1088 200x200 RGB images already divided into train and test sets. We are going to have a look to different classification techniques, starting with SVMs with RBF Kernel and MLP using low feature as input, then moving to CNN seen in two different cases, with gray scale and color images as input, and then ending with transfer learning and fine tuning on ResNet18.

## 2 Results

First thing first, let's have a look at the data, we can see a sample for each class in Fig.(1a), and the distribution of the training data on Fig.(1b), the test data is equally distributed. We can pre-process the data normalizing each image color channel between 0 and 1.


(a) Classes

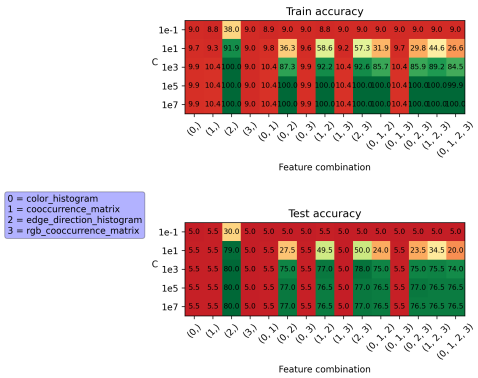



(b) Distribution Train Dataset

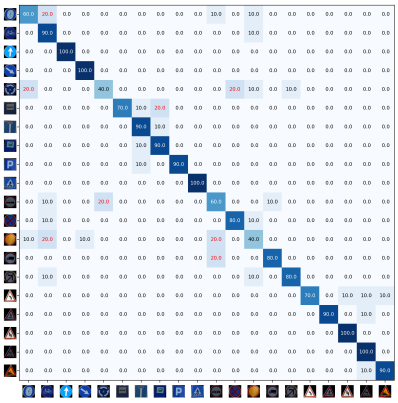
Figure 1

In order to do the training of SVM and MLP we need to extract low features from the training set, specifically these are 4: *color histogram*, *cooccurrence matrix*, *edge direction histogram*, *rgb*

*cooccurrence matrix*. We train the SVM with RBF kernel, with the approach *one vs one*, using the single and combinations of these low features and different $C$, that is the penalty parameter of the error term. It controls the trade off between smooth decision boundary and classifying the training points correctly. Fig.(2a) is very informative; it first tells us that low feature 2 is the



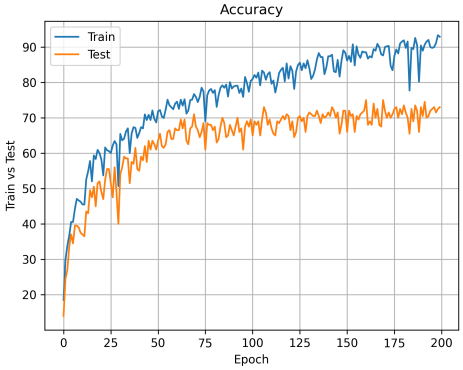

(a) RBF kernel SVM



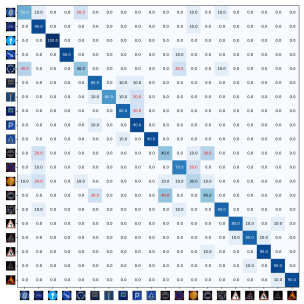
(b) Confusion Matrix

Figure 2

best representative of the images, and not only that, it is also the only one. We see how even combinations of low features gives lower performance to the SVM with respect to the single low feature 2. We can also see the confusion matrix of the best performing SVM Fig.(2b). Knowing the importance of low feature 2 we now turn to the training of the MLP, an attempt was made to train with various features but the behavior is that of the SVM, maximum accuracy on the test with low feature 2 and down with combinations of any other. That's why we focus on training with only that feature. The MLP have been designed to have 3 hidden layers $(512, 256, 128)$ and trained with batch, for 2000 epochs, a $lr = 0.001$ and Adam optimizer, that is a stochastic gradient descent method based on adaptive estimation of first-order and second-order moments.

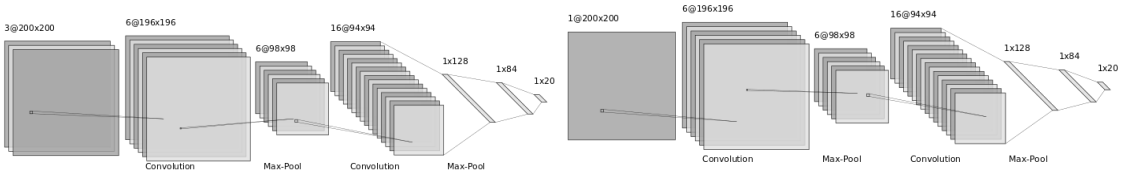



(a) MLP



(b) Confusion Matrix

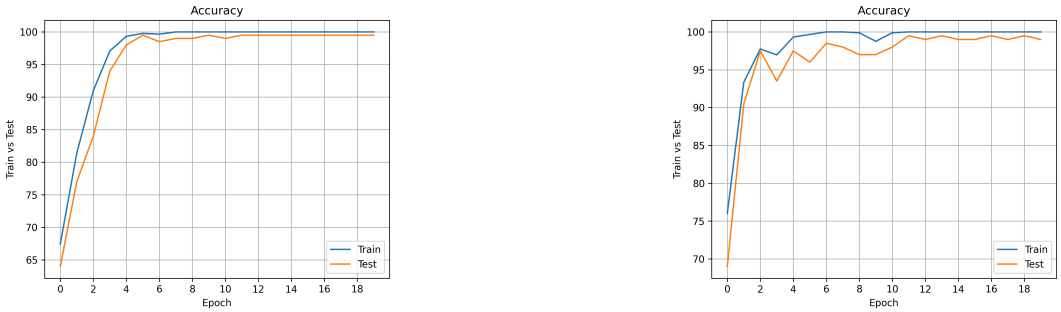
Figure 3




(a) ColorCNN (b) GrayCNN

Figure 4

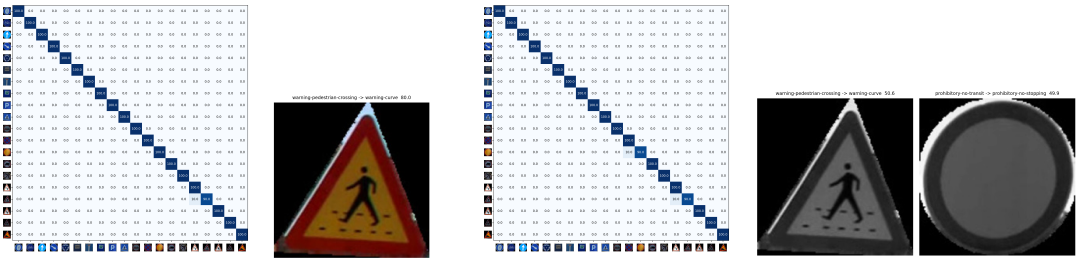
We can now move on to CNN, here in Fig.(4) we have the structure of the 2 CNN model whose inputs are color and grayscale images, respectively. Fig.(5), show us that the train of both CNNs is good, and most importantly they achieve very similar results in accuracy. It seems that in the short time, the grayscale CNN is faster to train but the color CNN, over multiple epochs, achieves the "best" result. The training, have been done batched, with Adam optimizer, a $lr = 0.0001$ and 20 epochs. They perform 99.5% and 99% accuracy on the test set.




(a) ColorCNN Train (b) GrayCNN Train

Figure 5

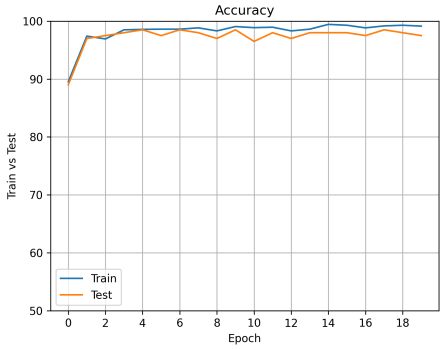
Here, in Fig.(6), the confusion matrix of both the CNNs. We can also see the miss classified sample, they are one for the ColorCNN and two for the GrayCNN. It is interesting to note that CNN's single error in color is with really high confidence, while CNN's grayscale errors are 2 but with much lower confidence.



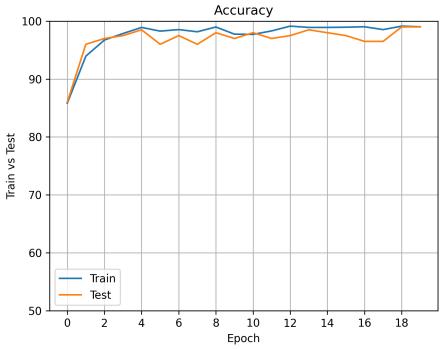
(a) ColorCNN CM (b) Color miss (c) GreyCNN CM (d) Grey miss

Figure 6

Since CNNs work so well, we can try to train them on augmented data in order to make them more robust Fig.(8b). Random Eraser, which involves randomly erasing a portion of the image, and Gaussian Noise were chosen to augment the data. We can see in Fig.(7) the CNNs trained on the new dataset, formed from the initial data with the addition of the version to which artifact was applied. The training, once again, have been done batched, with Adam optimizer, a $lr = 0.0001$ and 20 epochs. The accuracy is 98% and 93.5% respectively.




(a) ColorCNN AUG Train




(b) GrayCNN AUG Train

Figure 7

In this case, the color one works much more better then the gray scale one. The most common classes to be confused still, for the color one the *warning-pedestrian-crossing* and for the grey one we can have a look at Fig.(8a).

(a) GrayCNN miss


(b) Augmentation

Figure 8

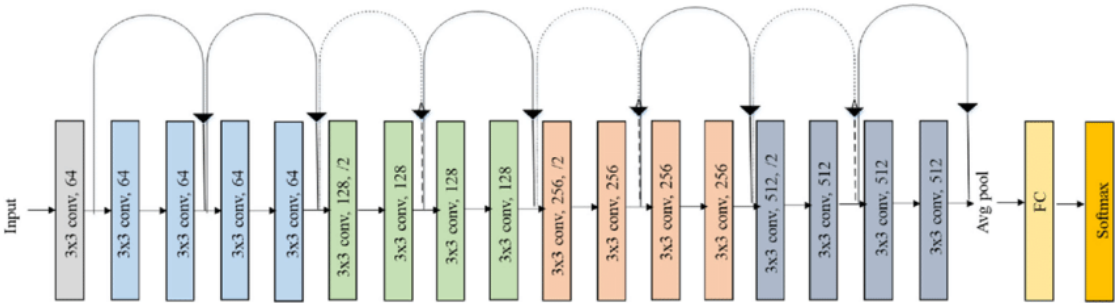



Figure 9: ResNet18

We can now try to do transfer learning on a pre-trained ResNet18, we can freeze the neural net weights and train only the weights of the last layer, the output layer. The last layer was trained batched, in 20 epochs, with Adam optimizer and scheduled learning rate, starting from 0.01 and exponential decreasing the learning rate every 5 epochs. We reach an accuracy, on the train set, of 95.35% and 100% on the test set. This strange behavior may be given by the too "simple" test dataset, in fact it does not contain any images to which artifacts have been applied.
Now, instead, we can try to fine tune a pre-trained ResNet18, this time we don't freeze the weights in order to let them change. The training have been done batched, with 10 epochs, Adam optimizer and scheduled learning rate, starting from 0.1 and exponential decreasing the learning rate every 5 epochs. We reach an accuracy of 99.66% on the train set and 100% on the test set.

I affirm that this report is the result of my own work and that I did not share any part of it with anyone else except the teacher.