Road signs classification
Machine Learning assignment
Marija Maneva

Abstract
Autonomous driving systems have revolutionized the automotive industry, paving the way for safer and more efficient transportation. One crucial component of such systems is accurately recognising and interpreting road signs. These signs provide essential information to drivers and play a vital role in ensuring road safety. A popular and effective model architecture is the Convolutional Neural Network (CNN) for detecting and recognising road signs in images. CNNs have proven to be highly successful in various computer vision tasks, including object detection and image classification.

This project aims to develop an autonomous driving system's road sign recognition module. It consists of analyzing the provided data, designing and implementing data pre-processing procedures, implementing and training a classification model and analyzing the behavior of the trained model using data processing and visualization techniques.

1. Data
The data consists of 1088 samples. In order to train and evaluate the company's system, they have selected 20 specific types of road signs. To ensure organized data management, the samples have been categorized into two directories for training and testing purposes, denoted as "train" and "test". Within both directories, the road-sign images are further categorized to have their own directory. The collected samples represent exemplars of each road-sign type, providing a diverse range of visual variations and environmental conditions.

Do you know how did a road-sign get promoted? It showed excellent "direction" and was always "sign-ificantly" visible! Blurry , vague and indistinct images are not always sign-ificantly visible to ML models. Let's help them to get that promotion with some data pre-processing.

1.1 Data pre-processing
To prepare the data for the training, it is necessary to extract the features from the images. As an additional operation, I incorporated normalization and histogram equalization into the feature extraction process.

- Normalization: the pixel values of the image are divided by 255 to normalize them between 0 and 1. This operation provides consistent scale, numerical stability, improved convergence, and enhanced feature extraction capabilities.
- Histogram equalization: the aim is to improve the quality and discriminative power of the extracted features by enhancing the contrast and distribution of the values in the array.

For feature extraction, I used :

- Low-level features: they are a set of predefined visual descriptors or characteristics extracted from images that capture basic and fundamental information about images after typically applying mathematical operations or filters. In this case I used "color_histogram" and "edge_direction_histogram" to gain the low-level features needed.
- Neural features: they are features extracted from neural networks obtained by passing input data (images) through the layers of a neural network and extracting the representations learned by the network.

In this case the neural networks are computed by using the "PVMLNet" CNN.

PVMLNet
PVMLNet is a modified form of the AlexNet architecture, designed to be simpler. It consists of eight convolutional layers and three fully-connected layers implemented as convolutions. After each layer, a

Rectified Linear Unit (ReLU) activation function is applied. The final output is passed through a softmax function to calculate the probabilities for each class. This CNN has been trained using the ILSVRC-12 subset of the ImageNet dataset.

And now , let's do some training …

2. Training and testing

The features extracted are trained with an MLP model. A Multi-Layer Perceptron (MLP) is a type of artificial neural network that consists of multiple layers of interconnected nodes, called neurons. MLPs are widely used for supervised learning tasks such as classification and regression.

Specifications used :

- *Number of batches = 50*
- *Learning rate = 0.001*
- *Steps = x.shape[0] // 50 (17)*
- *Number of epochs = 20000*

The results may be different by applying low-level and neural features.

How is this possible? Let's have a look at it …

2.1 Low-level features

Low-level feature extraction techniques aim to capture fundamental features from images, but they may not effectively capture intricate patterns or details. As a result, the accuracy of these techniques can be limited in scenarios that involve complex patterns.

What could possibly be the differences by using color images and grey-level images ?

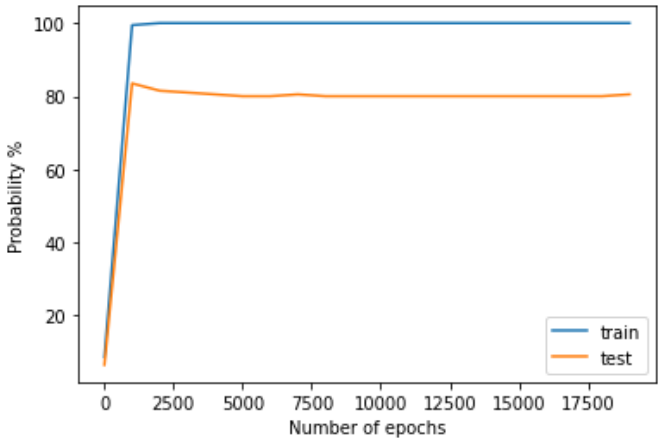

Figure 1 : train&test with low-level features (color_histogram&edge_direction_histogram)
color image

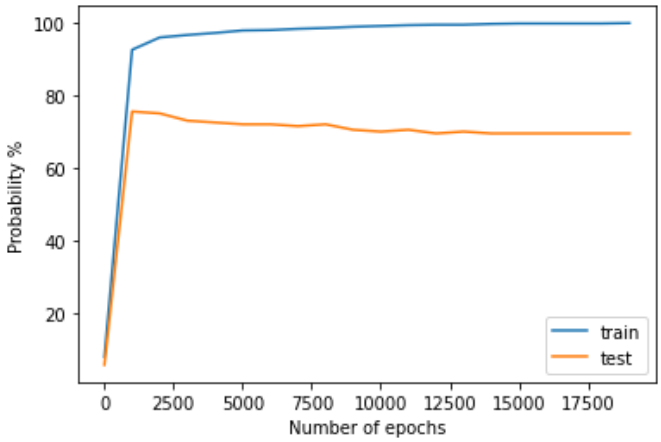

Figure 2 : train&test with low-level features (color_histogram&edge_direction_histogram)
grey-level image

Results obtained :

Type of image	Type of low-level feature	Train accuracy (%)	Test accuracy (%)
Color image	color + edge direction histogram	100%	80.5 %
Grey-level image	color + edge direction histogram	99.8 %	69.5 %

The highest accuracy reached is the one from color images. Colors can be used as a discriminative feature. Road signs often have specific color combinations or patterns that are indicative of their meaning or category. By using color images, the model can leverage this additional information to better distinguish between different types of road signs. As conclusion, we can say that the color cues can enhance the discriminative power of the model.

It's time to see what happens with training neural features …

2.2 Neural features

Neural features offer the benefit of capturing intricate and advanced information, enabling the model to potentially achieve superior performance.

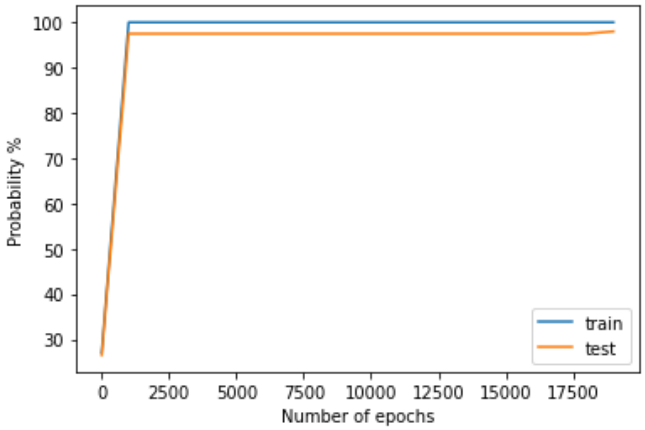

Figure 3 : train&test with neural features **color image**

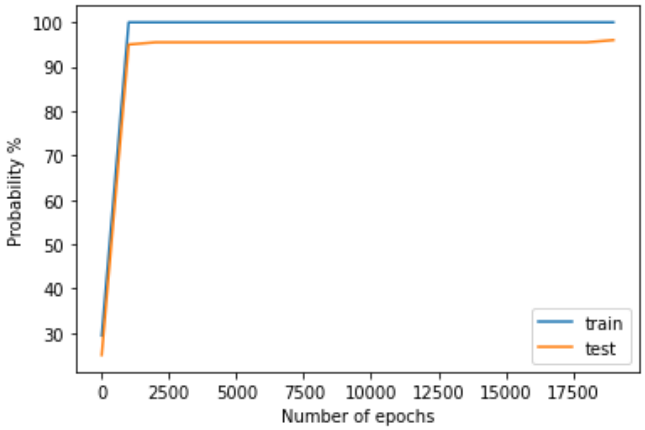

Figure 4 : train&test with neural features **grey-level image**

Results obtained :

Type of image	Train accuracy (%)	Test accuracy (%)
Color image	100%	98%
Grey-level image	100%	96%

Overall, the performance achieved with neural features is higher than the one obtained with the low-level features. As we have seen in the previous case, the accuracy with color images is slightly higher than the one with grey-level images for the same reasons explained above.

2.3 Transfer learning

The transfer learning process allows a target task model (CNN) to benefit from the knowledge coming from a pre-trained model (MLP).
The combination of the two models consists in putting the weights and biases from the MLP model into the last layer of the CNN model.

Results obtained :
The test accuracy reached in this case is 98 % for color images and 96% for grey-level images as it should be expected because transfer learning can speed up the training process, improve convergence, and boost the performance of the target task model.
By analyzing grey-level images, it's important to notice how without the histogram equalization operation the model is not able in the majority of the cases to recognise the sign on the image with the same efficiency that it does with it. The test accuracy increased from 88% (without equalization) to 96% (with equalization).

3. Analysis

Further analysis includes the computation of the confusion matrix and set up of a specific image to test with the model.
In this analysis section, I considered only the transfer learning process so the combination of the two models, MLP and CNN.

3.1 Confusion matrix

The confusion matrix is a useful tool for identifying potential misclassifications between different classes. By analyzing the entries that are not on the diagonal, we can pinpoint classes that are more likely to be misclassified. Higher values in these non-diagonal elements indicate a higher level of confusion between specific classes.

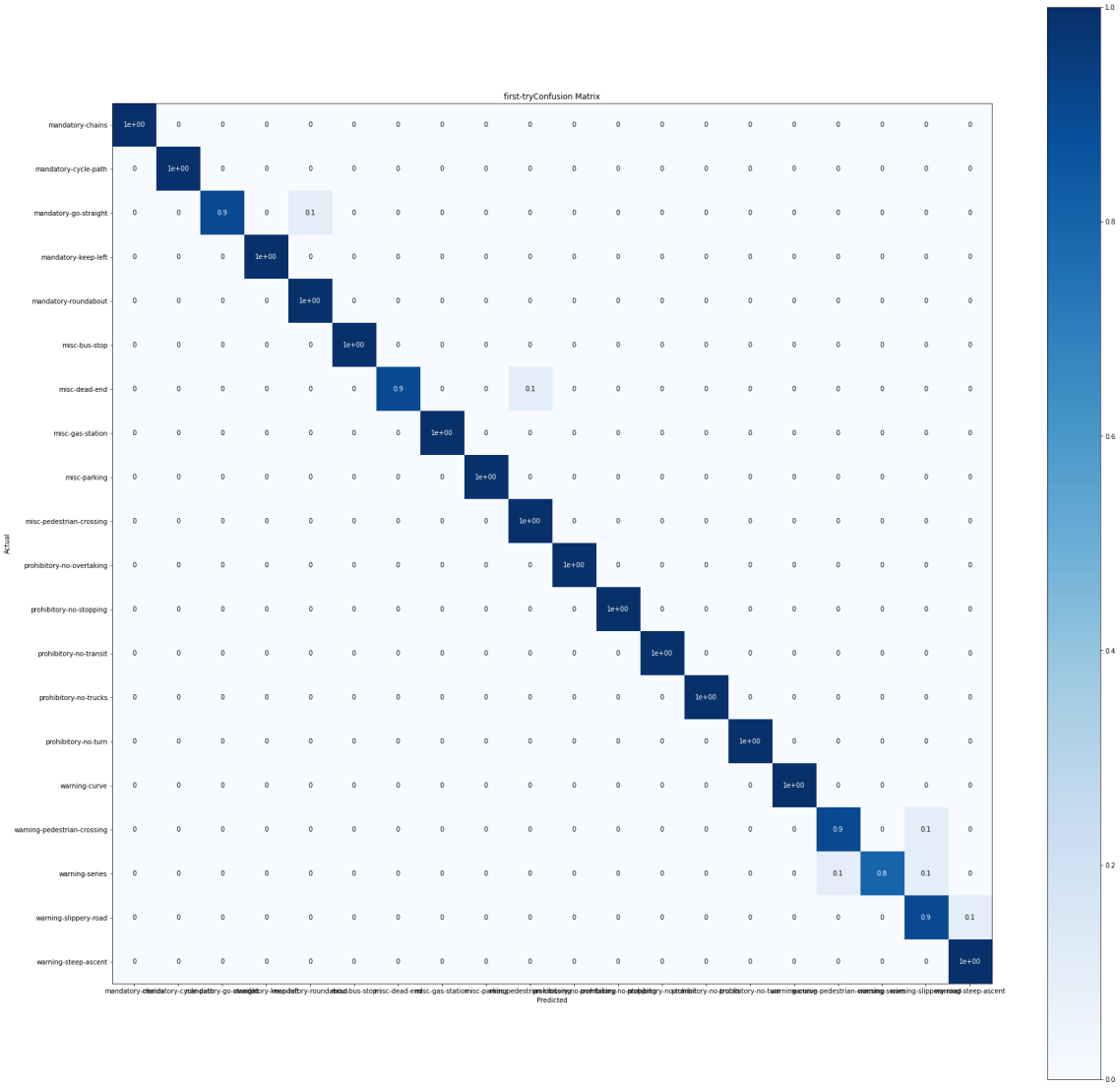

Figure 5 : confusion matrix on test set- color image

In the given case and also in the grey-level image case, the confusion matrix shows high values on the diagonal. This is considered desirable because it indicates that the model is effectively able to differentiate and accurately classify the various classes.

Now you're probably thinking: "all hat and no cattle". Be patient, we've come to the cattle part !
Now that we've seen the model's potential, let's put it in practice.
How would the model behave when you give in input an image to predict? Let's further test its ability …

3.1 Testing a specific road sign image

The process consists in loading an image and applying on it the combined model. The cnn.inference() function in the file script "transfer-learning.py" displays the top 5 predicted classes along with their corresponding probabilities. In this case, the parking road-sign has been analyzed and the model predicted with a 100% probability the right road-sign. By applying the same procedure to the same image, but in grey-level, the model predicted with almost the same efficiency (99.6 %) the road-sign.

Classes	Prediction (%)
parking	100%
pedestrian-crossing	0%
gas-station	0%
dead-end	0%

Figure 6 : display of the tested image (parking road-sign)

4. Conclusions

This project aimed to develop a road sign recognition module for an autonomous driving system using transfer learning. The combination of a pre-trained CNN model with an MLP model proved to be effective in improving the model's performance. The use of low-level features, such as color and edge direction histograms, provided reasonable accuracy, but the inclusion of neural features yielded superior results. Neural features captured more intricate patterns and details, leading to higher accuracy in road sign recognition. The combination of neural features, transfer learning, and appropriate preprocessing techniques resulted in a robust and accurate model capable of identifying and classifying road signs.

"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."