Deep Learning

Final Report

POLITÉCNICA

David Burrell, Ignacio Martínez, Miguel Pérez

03/04/2020

1. Introduction

This report presents the work done during the Deep Learning course concerning the development of multiple neural networks to classify different types of images. The work has been performed in Python utilising the Tensorflow and Keras libraries.

The document is structured in the following way. Section 2 describes the two datasets used, the German Traffic Sign Detection Benchmark and the CIFAR-100. Sections 3 and 4 describe the development of the neural networks from feedforward to convolutional architectures. These sections will also address two different types of task: classifying traffic signs and classifying general images. Section 5, on the other hand, focuses exclusively on the traffic signs. However, the task is not only concerned with the classification of traffic signs but also with their detection, by distinguishing them from the rest of the image. Finally, Section 6 ends with the main conclusions.

2. Datasets

2.1. German Traffic Sign Detection Benchmark

This dataset contains nine hundred images of pictures of German road signs. These are classified into 43 different types (see Figure 1). Each image can contain from zero to 6 traffic signs. Across the dataset the signs appear in a range of different lighting conditions and orientations.

Figure 1 *Examples of the 43 classes of the German Traffic Sign Detection Benchmark dataset.*

The dataset has been used both to classify isolated traffic signs, like the ones portrayed in Figure 1, and to identify traffic signs within a larger image.

2.2. CIFAR-100

This dataset contains 60,000 images divided into 100 different classes (see Figure 2). The classes are varied and range from animals and plants to household furniture and vehicles. The images in each class show the subject in a range of orientations and different lighting. This dataset, unlike the German traffic signs one, is composed of images centered on the object of interest.

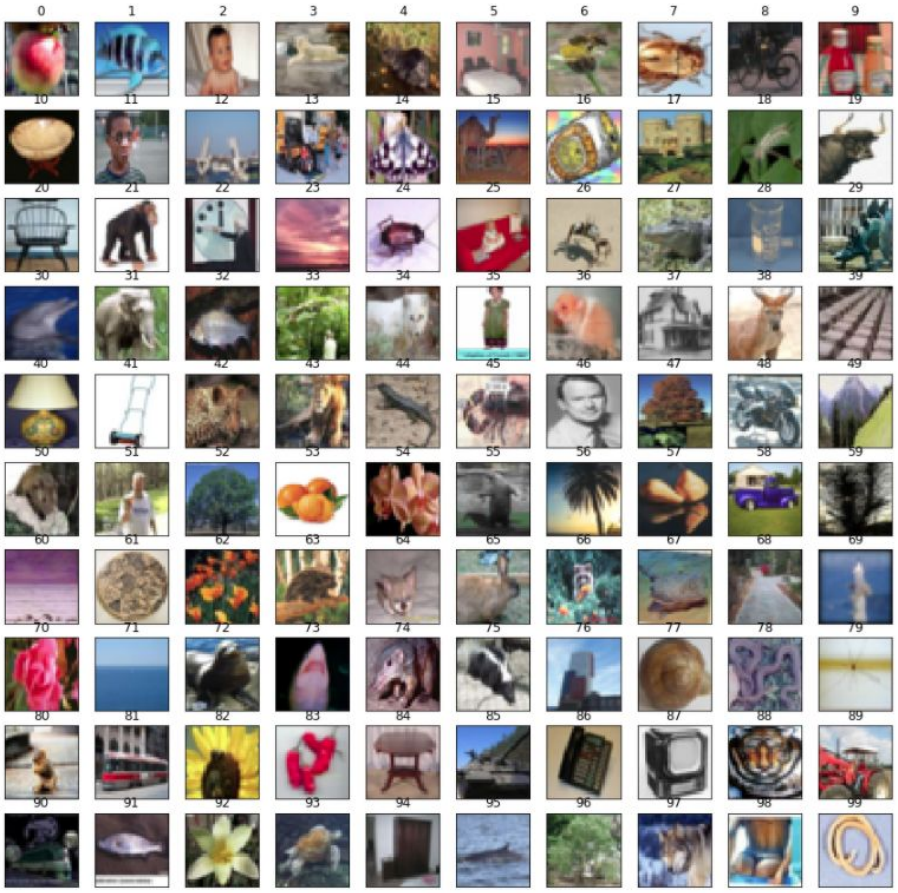

Figure 2 *Examples of the 100 classes of the CIFAR-100 dataset.*

3. Feedforward Neural Network

3.1. Network Architecture

The first neural network designed was a fully connected feedforward architecture. The same architecture (not including the output layer) was used for both classifying isolated traffic signs and to classify images of the CIFAR-100 dataset.

The architecture (see Figure 3) is composed of an input layer, two hidden layers with 1000 and 750 neurons respectively. The input layer and the two hidden layers have a dropout of 0.3 and use the ReLU activation function, which has provided the best accuracy values.

For the traffic sign dataset the output layer has 43 neurons, one for each of the traffic sign classes. The activation function chosen for the output layer is the softmax function. The network has been trained over 150 epochs, using a batch size of 25.

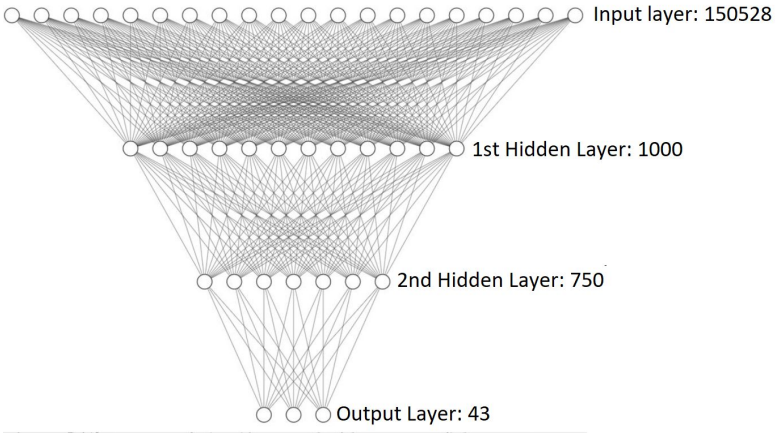

Figure 3 *Diagram of the feedforward network architecture.*

The optimizer chosen for the network is stochastic gradient descent optimizer (SGD). With hyperparameters values tuned to: learning rate = 0.0001, decay = 1e-6 and Nesterov momentum = 0.99. Other optimizers, like Adam, have been tried but yielded worse results.

For the CIFAR-100 dataset the network's output layer was adjusted to have 100 neurons to match the number of classes. Then, the best results were gained by reducing the epoch count to 100, while increasing the batch size to 32.

3.2. Results

For the German traffic signs dataset, the network achieved an accuracy of 0.9086 and a loss of 0.4035, as shown in Figure 4. The testing phase has taken 0.1964 seconds.

Figure 4 *Accuracy and loss (categorical cross entropy) results for the feedforward network with the training and validation datasets of the German road signs dataset.*

For the CIFAR-100, the test accuracy achieved was 0.2825 with a loss 3.0079, as shown in Figure 5 The testing process has taken 0.5459 seconds.

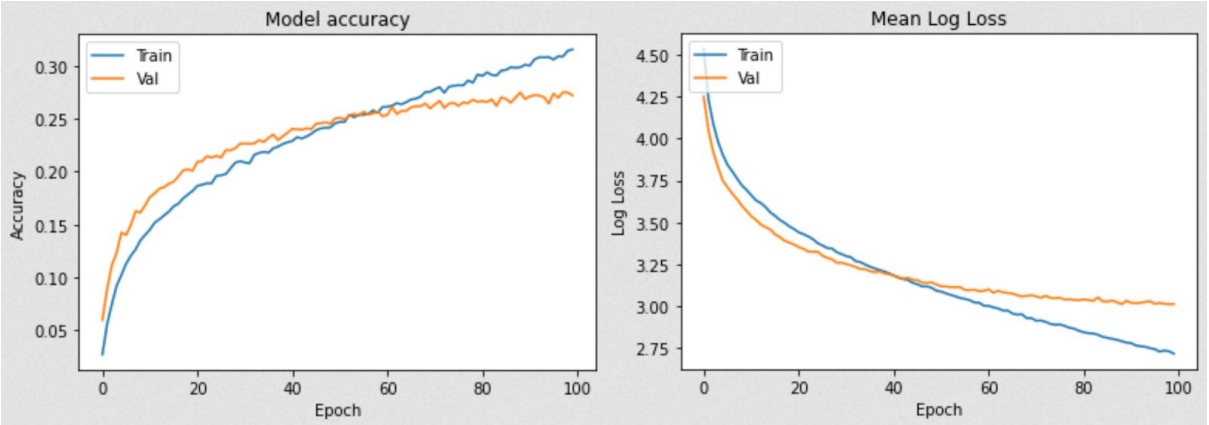

Figure 5 *Accuracy and loss (categorical cross entropy) results for the feedforward network with the training and validation datasets of the CIFAR-100 dataset.*

4. Convolutional Neural Network

4.1. Network Architecture

In this case, we have used two different strategies for the network architecture, one for the traffic sign dataset and another one for the CIFAR-100 dataset. Convolutional layers are used on both.

4.1.1. German Traffic Signs - Single CNN

Several approaches have been tried for the road signs dataset. The first one was a convolutional neural network. It has been inspired by AlexNet and is made of four convolutional hidden layers, followed by one fully connected hidden layer and, finally, an output layer. The first two convolutional layers use max-pooling and all layers use the ReLU activation function, except for the output layer which uses the softmax function. The fully connected hidden layer has a dropout value of 0.8. We will be referring to this architecture as Model A.

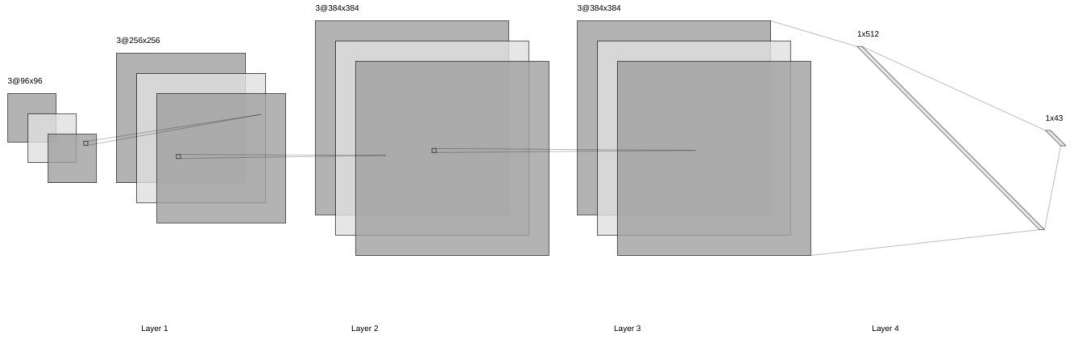

Figure 6 *Network architecture of Model A.*

The optimizer chosen for Model A is the SGD, with a learning rate of 0.0075, a decay = 1e-6, and Nesterov momentum of 0.9. The network has been trained with 200 epochs, using a batch size of 150.

4.1.2. German Traffic Signs - Single CNN and Data Augmentation

Data augmentation has been tried to improve the results obtained by Model A. The theory is that if the training images are modified in a correct manner, the increased variety and number of images in the training dataset helps the neural network to be more prepared to face the unseen testing dataset.

In an attempt to increase the results, the following modifications were made to the training dataset:

- width_shift_range = 0.1
- height_shift_range = 0.1
- zoom_range = 0.2
- shear_range = 0.1
- rotation_range = 10.0

The Model A has been trained with this augmented training set during 30 epochs with a batch size of 150.

4.1.3. German Traffic Signs - Majority Vote

One second attempt has been tried to improve the results based on Model A, this time using a majority vote process. The process consists of three neural networks, each classifying an image. Then, the results of each are compared and whichever class has the most voters between the three is assigned to the image.

For our first model we used Model A as described above. Then for the second we changed Model A to use LeakyReLU activation functions instead of the ReLU and modified the number of filters, we call this new model Model B. Then for the final model we again modified Model A to use LeakyReLu, but only to use the first three convolutional layers and a larger fully connected hidden layer (increase the neuron count from 512 to 1024). All three models have been compiled with a SGD optimizer with a learning rate of 0.0075, a decay of 1e-6 and a Nesterov momentum of 0.9. They have been trained during 200 epochs with a batch size of 150.

4.1.4. CIFAR-100 - Single CNN

For the CIFAR-100 dataset we have decided to use a different architecture, not based on Model A. This architecture is composed of six hidden convolutional layers and one hidden fully connected layer. They all use the ELU activation function and all layers but the first, third and fifth use a dropout of 0.2 and max-pooling with a pool size of 2x2. The full model is shown in Figure 7.

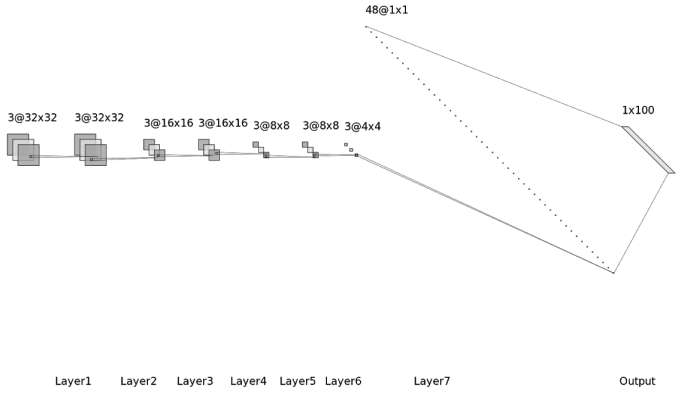

Figure 7 *Architecture of the convolutional neural network designed for the CIFAR-100 dataset.*

The optimizer chosen for the network is the Adam, with a learning rate of 0.0001. The network has been trained with 25 epochs and a batch size of 16.

4.1.5. CIFAR-100 - Single CNN and Data Augmentation

This architecture has also been trained with an augmented dataset, following the approach followed for the road signs dataset. The following modifications have been made to the training dataset:

- width_shift_range = 0.1
- height_shift_range = 0.1
- shear_range = 0.1
- zoom_range = 0.1
- fill_mode = 'nearest'

The network has been trained with data augmentation during 20 epochs, with a batch size of 16.

4.2. Results

Of all the approaches tried for the road signs dataset, Model A has been the one that performed the best. The Model A has achieved, in 4.35 seconds, an accuracy of 0.9806 and a loss of 0.1589 (see Figure 8). The Model A trained with the augmented training dataset achieved the same accuracy but had a slightly bigger loss value (0.1758). The majority vote approach yielded the worst results, with an accuracy of 0.9612 and a loss of 0.2293.

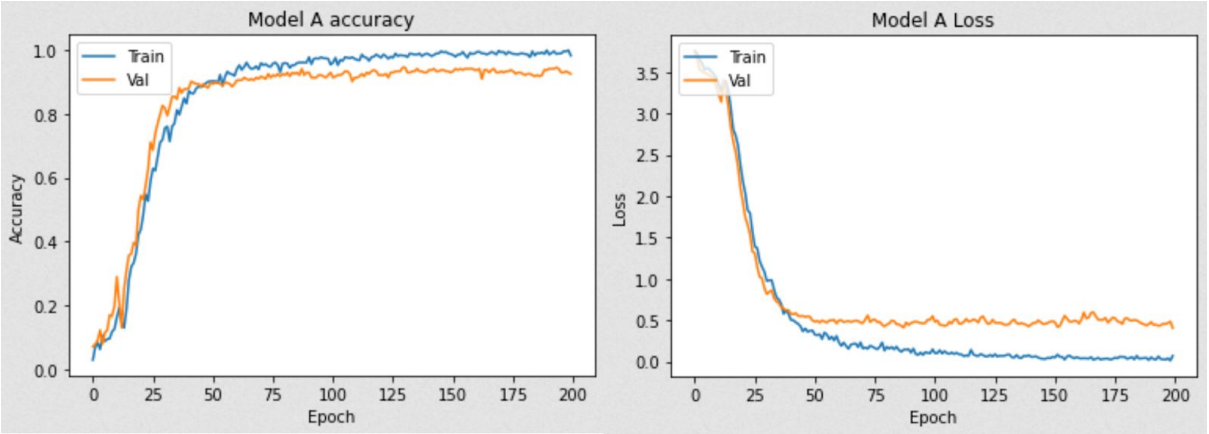

Figure 8 *Accuracy and loss (categorical cross entropy) results for the Model A network with the training and validation datasets of the German road signs dataset.*

The results for the CIFAR-100 testing dataset have an accuracy of 0.503 and a loss of 2.1259, and the testing process has taken 1.04412 seconds (see Figure 9). The accuracy results for the augmented dataset have been the same, but the loss is slightly lower: 2.04142.

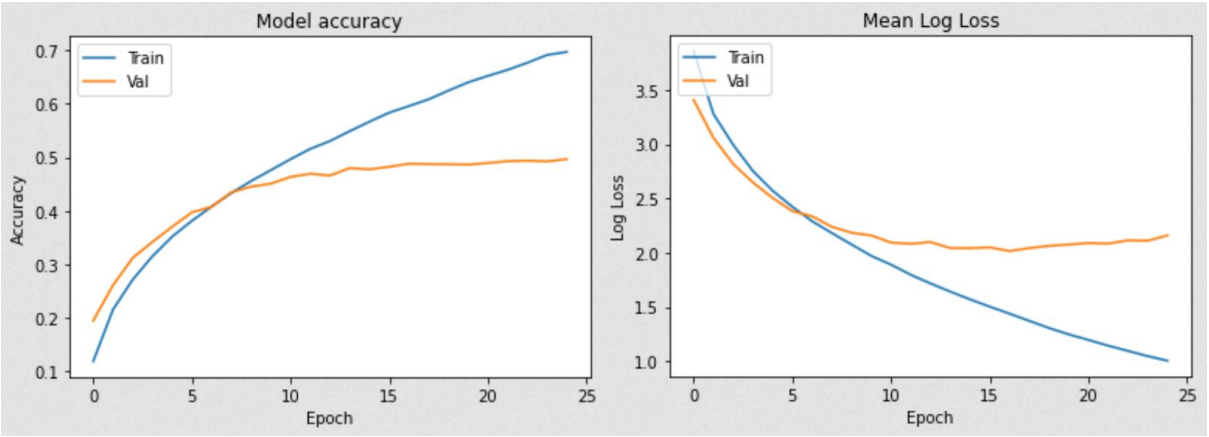

Figure 9 *Accuracy and loss (categorical cross entropy) results for the convolutional network with the training and validation datasets of the CIFAR-100 dataset.*

5. Traffic Sign Detection

5.1. Method Architecture

For this task rather than having to classify an image as belonging to a certain class of traffic sign, we were required first to locate possible locations then assign the signs a class, as described in Section 2.2. To achieve this, we were required to build more than a neural network, but an image processing pipeline that could be used to identify possible candidates then classify them. It should be noted that having attempted with a previous model that did not provide desirable results we took inspiration from others class groups that took different approaches.

To start this task we were provided a Region Proposal Network (RPN) that used a pretrained ResNet model backbone. This RPN was then used to generate the proposed location of signs in an image. Proposed images were given by x_1,y_1 and x_2,y_2 coordinates making a bounding box. It was found that

the RPN gave the best set of proposed locations when tuned to find 9500 locations at first, then after it performed non-maximum suppression (NMS) to return a maximum of 2500 locations per image. The NSM was set to use an overlap threshold of 0.5. To determine if the final proposed locations were in fact a sign, and what class they were, a 5 stage processing pipeline was developed:

1. Remove boxes near the vertical extremes of the images
2. Remove candidate boxes with edges of a "large" size
3. Remove candidate boxes with a "rectangular" aspect ratio
4. Use a CNN to classify if the image area is a sign or background scenery
5. Use a CNN to classify the type of sign

Below is a description of the rationale behind each step and how they were implemented

5.1.1 Extreme vertical edges removal

It was noticed that in the images signs did not appear at the top or bottom of the as shown in Figure 10 where the traffic signs are marked with a blue box.

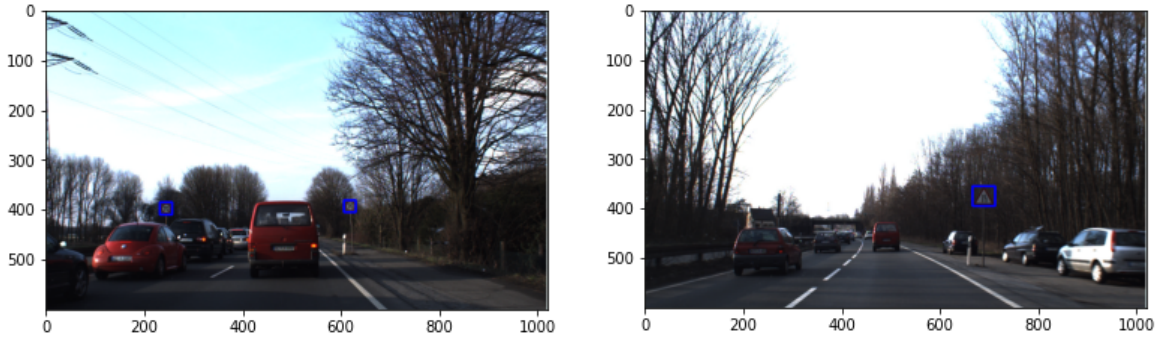

Figure 10 *Two examples of images with the road signs located inside a blue box. It can be seen that these types of pictures do not have signs near the top or bottom of the image.*

This allows for the filtering of all bounding boxes that lay close to the minimum and maximum of the y axis of an image. It was found by setting this filter to remove all bounding boxes with a y_2 value less than 50 and greater than 600. A common false positive this removed was the sun or camera glare which, as they are round bright shapes, were commonly classified as being a sign.

5.1.2 "Large" box edges

Within the images, signs take up a relatively small area (see Figure 10). Thus it is possible to filter out boxes that are of a certain size in both width or height. Upon experimenting with different values it was found that the best results were given when removing boces with a width greater than 250 or a height greater than 200.

5.1.3 "Rectangular" aspect ratio removal

Similarly to the required boxes having a certain size they also were generally square in shape, due to the signs being circles and isosceles triangles. To be able to filter on this the aspect ratio was calculated for each proposed bounding box. It was found that the best results were achieved when

boxes with a ratio less than 0.8 and greater than 1.2 were removed, which is logical due to the aspect ratio of a square being 1.

5.1.4 CNN to classify if the image area is a sign

Once the filtering of boxes due to their appearance is completed, it is possible to use convolutional neural networks to filter on the actual content of the boxes. The first network used does not attempt to classify the type of sign but only if the proposed location is an image or not. This CNN was built using a pre-trained ResNet model with an additional global average pooling layer added, with a dropout rate of 0.7, and an output layer with 2 neurons (for binary classification for is a sign or not) with softmax activation. An SGD optimizer was used with a learning rate of 0.001, a decay of 1e-6 and of momentum = 0.99. This model was originally developed for the next step but was found to be sufficient for both tasks.

To train the model a new dataset needed to be created that consisted of images that were not of a sign, classified as 0, and those that were of a sign, classified as 1. There was no need to generate the sign images as it was possible to repurpose the "sign type" data set and change the label to 1 for all images. To generate the 0 class images a randomly generated set of 852 images were taken from the test images that were 100x100 pixels. It was ensured these would not overlap with any of the sign locations by only using the top and bottom sections of an image for sampling. A selection of the images generated during this process can be seen below in Figure 11.

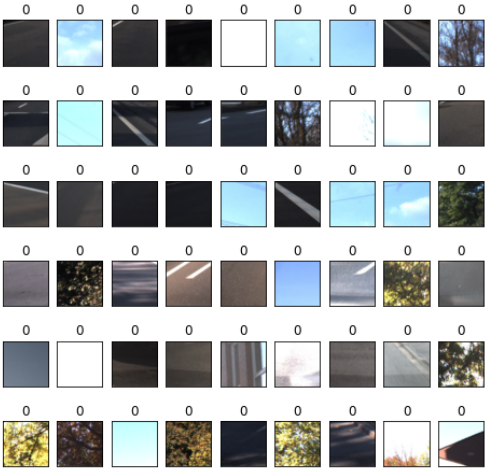

Figure 11 *Examples of background images used to train the background detector.*

After an image had been classified by this CNN it was only kept if it had been classified as a sign (label 1) with a confidence of 70% or higher.

5.1.5 CNN to classify the type of sign

The final step in the pipeline is actually to classify the images as a type of sign. As previously mentioned in the above section this was done using a pretrained REsNet model. The only change is the number of layers in the output layer were increased to 43 to represent the different class signs. The final set of images said to be signs were only kept if the final CNN has a confidence of 70% or higher, otherwise they were discarded.

5.2. Results

Rather than using accuracy of classification to evaluate this process the mean average precision (mAP) metric was used. Rather than only using the signs that were classified correctly / not correctly (or not classified at all as they were filtered out) it also takes into consideration the images that were classified but were not actually signs at all. This solution was able to get an average score of 48.19% across all sign classes when processing 299 images, in 918.65 second (approximately 3 seconds an image). Although not a very high value, as can be seen from the Table 1, the system was able to perform very well on some sign types, several even achieving 100%.

Table 1 *mAP score for the different sign classes.*

Sign Type	**mAP (%)**
1	62.39
2	61.96
3	42.22
4	55.32
5	35.62
6	75
7	75
8	67.14
9	66.67
10	57.25
11	63.46
12	71.54
13	76.34
14	85.32
15	60.00
16	100
17	75
18	45.45
19	0

20	0
21	0
22	50
23	79.22
24	0
25	27.50
26	14.29
27	0
28	0
29	0
30	100
31	0
32	20
33	100
34	66.67
35	80
36	0
37	0
38	49.11
39	0
40	0
41	0
42	68.75

6. Conclusions

It is notable that the evolution in structure also provided us an increase in the accuracy score as well as the time for training. We parted from general purpose machine learning approach to more specific image recognition tasks in order to track the progress in the performance. We have also realized

during the training of the deep learning networks that trial and error is an important part of such training, once you have the main basic ideas of your project already established. Also, we can say that increasing the depth of the network is not always going to provide better results, but sometimes it does.

For the last assignment, the traffic sign detection, we applied a more realistic approach to get the data from a real world image source (such as dash cams from vehicles), which helped us to realize that in the Deep Learning environment (as well as in the ML) the data acquisition and data augmentation is as relevant as the training and model building part.