# Multiple traffic sign detection project report

Bojana Počuča Marco Nobile

## I. Introduction

Template matching is a common problem in Computer Vision where an algorithm is trying to find similarities between one or more *template images* and the *source/target image*. This problem is known to be one of the most complex due to potential deformations in the source image. Nowadays, it is possible to deal with these challenges in a much faster, more robust manner with data driven approaches such as Deep Learning algorithms, that learn progressively more about the image as it goes through each neural network layer. Since template detection is built upon feature extraction, a logical solution to the problem would be implementing one of the most prominent Deep Learning techniques for working with images - *Convolutional Neural Networks* or **CNN** which are known to be the most powerful approach for image processing, classification and segmentation. The report is structured as follows: in *Part II* we explain how we used transfer learning on a CNN to obtain a feature extractor, then we describe a possible solution to the template detection problem with a *Fully Convolutional Network* or **FCN**. Subsequently, the architecture of these 2 combined networks would be studied along with the overview of their differences, pros and cons. In *Part III*, results will be shown along with the corresponding experiment information and the report will be concluded in *Part IV*. All things considered, we obtain good results in a sparse template setting.

## II. Proposed approach

Firstly, in order not to start the learning from scratch, we adopt a *transfer learning* approach, using a pre-trained **VGG-16** model that was trained on a benchmark dataset to solve a classification problem [1]. In the original VGG model there are convolutional layers and fully connected (FC) layers, and the main idea in transfer learning is to remove the original trained classifier (FC layer) and replace it with a new instance of another FC layer(s) that are going to learn the new task at hand. By training this model on our images we get as output a probability distribution over the possible templates for each input.

The next step is to move from a coarse inference to a fine inference by making a prediction at every pixel. By casting these fully connected layers into 1x1 convolutions we achieve a pixel level prediction, retrieving a network architecture that takes as input an image of any size and give us classification map as output [2] which is only 8 times smaller then the input image. This process is the so called 'convolutionalization' of classic CNN, where instead of a fully connected layer with fixed-length feature vectors for classification, we use de-convolution layers (or the *up-sampling* layers) to retrieve a 2D-classification map as output. This architecture is shown in Fig. 1. Here the feature map is obtained by up-sampling the 1x1 convolutions outputs in order to retrieve a dense feature map that segments the input image. With this technique a prediction can be generated for each pixel, while retaining the spatial information in the original input image [3]. The idea from the paper we followed [2] was to combine the convolved fine layers (pool3, pool4) with the convolved coarse layers (pool5) of the network in order for the model to make local predictions that respect global structures (see Fig. 2).

## III. Experiments and results

In this section we are going to present the results that we achieved by using the above described techniques.

*a) Datasets:* The data set we used is the **GTSDB** [4], in which there are 43 unique templates, of which we have 1213 examples. In the same data set we are provided with 900 different pictures of street views that include different traffic signs.

*b) Fine-tuning VGG:* Before reaching a pixel-level prediction we first had to fine-tune a VGG-16 [5], that has been provided in the Torchvision module of **Pytorch** [6]. In this initial task our goal was to classify the templates and the background correctly, in order to obtain solid convolutional layers which will later be used as feature extractors in the FCN model. We have been able to load the original parameters of the VGG architecture and by training the whole network with a very low learning rate (*lr = 1e-5*), we have managed to achieve *97.73%* and *62.34%* of accuracy on the templates used in the train and validation set respectively. The output of this CNN classifier can be seen on Fig. 3. Given the low amount of initial observations, we have used some *data augmenting* techniques [7] and *weight decay (wd = 5e-4)* to avoid overfitting on the training data as well as to achieve a better accuracy on diverse input images.

*c) Training FCN:* Next, we have proceeded onto building and training the *FCN-8s* [2] described above. In order to do so we have built for each street picture the relative ground truth, using the annotations provided with the dataset. One of our main concerns has been the option to build these labels in a dynamic manner, such that we can freely train on any chosen input resolution. This has been fundamental since we had to tune our batch size (*bs = 20*) in conjunction to the input resolution (*512x870* instead of the native *800x1360*), to be able to avoid any memory issues on the GPU. Training for 25 epochs on the whole dataset (16 hours) we have been able to retrieve some initial qualitative results. As shown in Fig. 5 the templates of the target image were detected correctly (we can see that 2 of the same templates have a different color from the third, different template), and we have obtained good results on both medium and high resolution pictures, while for lower resolutions we could not achieve any meaningful results. The pixel accuracy can be found in Table. I, where we present for different input resolutions the scores we obtained for pixel-wise template detection and pixel-wise template classification.

## IV. Conclusions

All in all, a good result was achieved in detecting *where* the templates are, but further improvements are still necessary with classifying exactly *which* template was detected. We suppose that this is due to lack of training/data in both the networks (CNN and FCN). In addition to this, with more computational power and training time on high resolution images and with a proper ground truth for a segmentation task our results could be improved. Nonetheless we have been able to detect silhouettes of templates consistently, and given the difficult setting for a segmentation task (i.e. very small templates in the target images), we can say that we are content with our results.

| | Input Resolution | |
|---|---|---|
| Pixel accuracy for: | 512x870 | 800x1360 |
| Classification | 23.38% | 22.31% |
| Detection | 44.12% | 64.41% |

TABLE I
PIXEL ACCURACY FOR DIFFERENT INPUT RESOLUTIONS AND EVALUATIONS

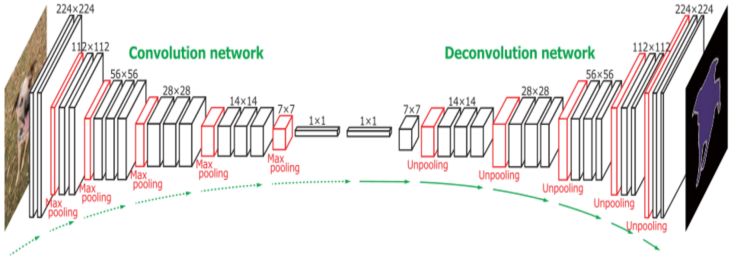


Fig. 1. Network Architecture

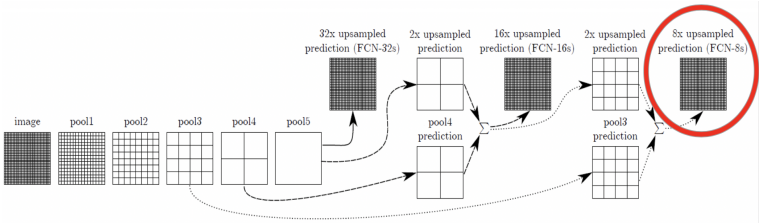


Fig. 2. FCN-8s


Fig. 3. Example of templates and background class (43)

Fig. 4. Street View

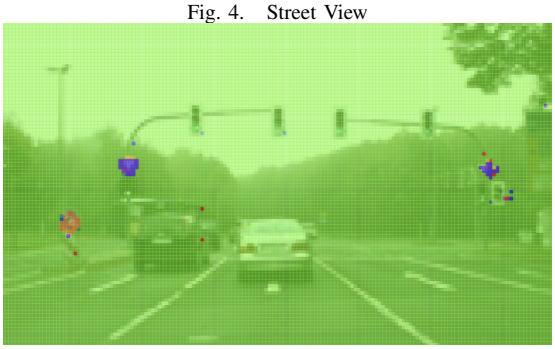

Fig. 5. Template detection

## REFERENCES


[1] Title: Transfer learning from pre-trained models
Autor(s): Pedro Marcelino
Link: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

[2] Title: Fully Convolutional Networks for Semantic Segmentation
Author(s): Jonathan Long, Evan Shelhamer, Trevor Darrell
Link: https://arxiv.org/abs/1411.4038

[3] Title: Explain CNN and FCN
Autor(s): Arya
Link: https://zhuanlan.zhihu.com/p/65315091

[4] Title: German Traffic Sign Detection Benchmark GTSDB
Author(s): Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, Christian Igel
Link: https://erda.ku.dk/public/archives/ff17dc924eba88d5d01a807357d6614c/published-archive.html

[5] Title: Very Deep Convolutional Networks for Large-Scale Image Recognition
Author(s): Karen Simonyan, Andrew Zisserman
Link: https://arxiv.org/abs/1409.1556

[6] Title: Transfer Learning for Computer Vision
Author(s): Sasank Chilamkurthy
Link: https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

[7] Title: Imgaug package
Author(s): Jung, Alexander B., Wada, Kentaro, Crall, Jon and others
Link: https://github.com/aleju/imgaug