Bulletin of the *Transilvania* University of Braşov • Vol. 10 (59) No. 2 - 2017
Series I: Engineering Sciences

ROAD RECOGNITION USING FULLY CONVOLUTIONAL NEURAL NETWORKS

E. HORVÁTH[1] C. POZNA[2] Á. BALLAGI[3]

Abstract: *In the past few years an increasing tendency is perceptible in the field of research and application of neural networks and deep learning. For image classification and segmentation convolutional neural networks are used intensively. Our goal is to solve the image recognition subsystem of a self-driving vehicle, which will take part in a competition. In order to fulfil this task fully convolutional neural networks were used. The present paper introduces an early-stage work for road recognition. In the current work-in-progress paper, the results are presented and further developments are discussed.*

Key words: *deep learning, convolutional neural networks, semantic segmentation.*

1. Introduction

The present paper focuses on the image recognition subsystem of a race vehicle. Our goal is to take part in the Shell Eco-marathon Self-driving challenge with our university's team. The Shell Eco-marathon is originally a fuel saving competition; the self-driving category will be introduced in the next year, in 2018. There are of course numerous sub-tasks regarding the self-driving tasks, but the robust road detection is one of the key elements, and this paper only targets this domain.

The paper is organised as follows. This section gives a quick introduction, and necessary knowledge about the work. The 2nd section deals with the problem definition, we present the scope of the work and what is outside from the scope. This section describes also the existing neural network architectures for classification and segmentation. The 3rd section describes the issues regarding the implementation. The 4th section describes the first results and the final section deals with possible future improvements.

2. Problem Definition

As mentioned, this paper focuses on image recognition, especially road detection. The naïve way of road detection would be only to use traditional computer vision (CV) algorithms e.g. colour segmentation, region-growing methods or edge detection.

[1] Dept. of Computer Engineering, *Széchenyi István* University, Győr.
[2] Dept. of Automatics, *Transilvania* University of Braşov.
[3] Dept. of Automation, *Széchenyi István* University, Győr.

These algorithms do not perform well in our case [5]. The reasons for that are the lightning conditions, the object's perspectives and sizes vary significantly, see Figure 1. Karpathy et al. gathered all the challenges of recognizing a visual concept, as follows [10]:

- Scale variation: Visual classes often show variation in their size: This means size in the real world, not only in terms of their amount in the image [10].
- Viewpoint variation: A single instance of an object can be oriented in many ways with respect to the camera [10].
- Deformation: Many objects of interest are not rigid bodies and can be deformed in extreme ways [10].
- Occlusion: The objects of interest can be occluded. Sometimes only a small portion of an object (as little as few pixels) could be visible.
- Illumination conditions. The effects of illumination are drastic on the pixel level [10].
- Background clutter. The objects may blend into their environment, making them hard to identify [10].
- Intra-class variation. The classes of interest can often be relatively broad, such as car. There are many different types of these objects, each with their own appearance [10].

The other approach to explain why the classical CV algorithms won't work is related to the Moravec's paradox. Moravec's paradox is the means that, opposing to previous expectations, high-level reasoning usually requires little computation, but low-level skills, which are usually unconscious require enormous computational resources [5], [6].

Fig. 1. *Typical images of the competition*

Besides that, the vehicles on the road are very ihomogen regarding the shape, the size or the colour, see Figure 2. This is a concern because the first task which is road segmentation is also reliant to the vehicle segmentation.

Fig. 2. *Inhomogen vehicles in the competition*

Considering these facts we decided to use neural network approach (NN) instead of the traditional CV approach. In the next section the possibilities, and the existing soluions will be examined.

2.1. Existing Neural Network Architectures for Classification and Segmentation

There are many architectures for the *classification* problem. To name a few well known: the VGG's networks (Visual Geometry Group, University of Oxford) [8], AlexNet [9], and GoogLeNet [10]. These algorithms focus on classifying objects from images. VGG's approach is based on increased depth networks (16-19 weight layers) using an architecture with very small (3×3) convolution filters, with a stride of one pixel. The results show that there is a significant improvement on the prior-art configurations by pushing this approach [8]. In contrast, AlexNet uses not only (3×3) convolution filters, but also a relatively large kernel size (11×11), with a stride of 4 pixels [9]. GoogLeNet uses (3×3), (5×5) and (7×7) kernel sizes too. Kernel size is one of the many parameters of the architecture. There are differences in convolutional layer quantity as well: AlexNet uses 5, VGG uses 13 to 16, and GoogLeNet uses 22 layers. Besides that, all of them have at least 3 fully connected layers, several pooling layers and softmax in different layouts.

To summarize, AlexNet is one of the first successful convolutional neural networks, with a relatively small and simple architecture. VGG is much deeper architecture and uses uniform kernel size so it can be considered a large, but simple architecture. GoogLeNet has a quite complex layout and a lack of sequential structure. All of the mentioned solutions composite the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) with success. To mention a few results AlexNet architecture won ILSVRC12, VGG finished as 2nd ILSVRC14 and the GoogLeNet won the same year. Of course, these competitions are not the only way to measure the vitality of these architectures in dissimilar the result may change.

The *segmentation* problem is also intensively-researched, so many NN architectures are present, and most of them rely on partly of fully convolutional neural networks (FCN). In contrast to classification where the whole image is treated as one unit, segmentation attempts to classify pixel-wise into one of the pre-determined classes. To name some of the well-known architectures: the BVLC FCN [9] and it's variables (FCN32-s, FCN16-s, FCN8-s), SegNet [2] and the mentioned VGG network can also be used for this task too with an extension of upsampling layers [9]. VGG-VD [8] model is trained to perform classification in the ImageNet Large Scale Visual Recognition Challenge data, and the pre-trained model is available to download. A training took 2-3 weeks on 4 NVIDIA Titan Black GPUs [8]. VGG-VD16 is a 16-weight layer variant of the VGG architecture, which contains 13 convolutional layers and 3 fully-connected layers. Respectively VGG-VD19 contains 19 weight layers, 16 convolutional layers and 3 fully-connected layers.

3. Implementation

The available computer for the task was equipped with Intel Core i7-4820K Processor, 10M Cache, 3.90 GHz and 16 GB DDR3. Unfortunately, no suitable GPU was available at the time. The computer was running a Windows 10 operating system, which was not mandatory operating system over Linux but it was preferred for some reason. We examined the existing frameworks and software libraries for neural network implementation. At the time, the paper was written the most notable possibilities are TensorFlow, Caffe, Microsoft Cognitive Toolkit (CNTK), Keras, theano, torch and DL4J. We found that for our purposes TensorFlow is the most well-documented and flexible solution with a large community. The approach of TensorFlow is to represent numerical computation using dataflow graphs. The nodes in the dataflow graph represent mathematical operations, while the graph edges represent the tensors communicated between them [1].

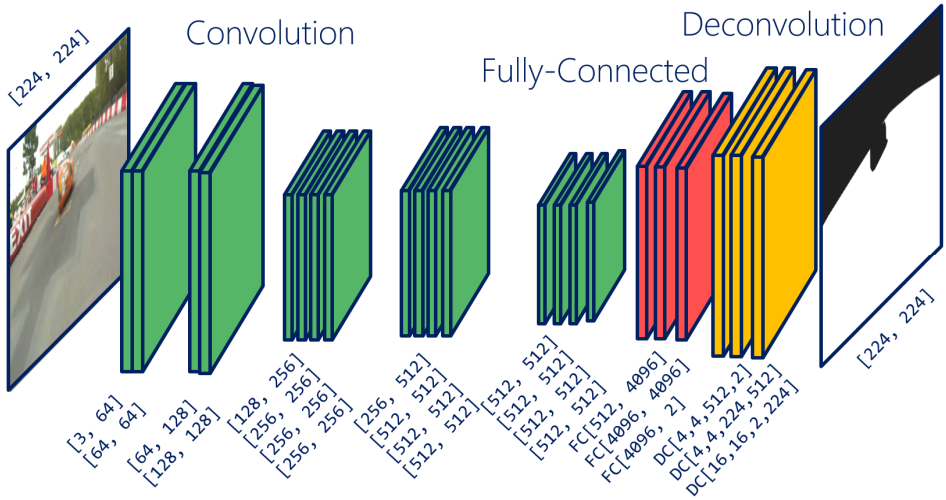

Fig. 3. *Visualisation of the used VGG-VD-19 layout*

The first step was the preparation of the images. The training, validation and the test set is a mixture of own annotated images and images from public image repositories, in a ratio of 60% and 40% respectively.

The programming was done in Python, which is the most common API to access TensorFlow. There are several publically available codes for similar tasks [1], [7], [12] all of them assume Unix-based operating systems. Considering the pros and cons of the previous section the VGG-VD19 architecture was chosen as our model with an addition of deconvolution layers for upsampling. The model is a 19 layer deep network with convolutional layers, max pooling and with only 3×3 convolution kernel size. This simple yet efficient model may easily transfer into other implementations, which is our final goal. The latest solution uses the pre-trained VGG model and some parts of the mentioned publically available source codes. The architecture of the VGG-VD19 [7], [12] is represented on Figure 3.

4. Results and Experiences

For the sake of simplicity, we decided to use only two categories at our images, the first category is the road and the second is all other parts of the images. We annotated a small quantity of images from the Shell Eco-marathon race (350 images). We added 200 more images and annotations form various downloaded public image repositories. As expected, choosing a proper learning rate is beneficial for the learning. In Figure 4 cross-entropy loss of different learning rates for the Adam Optimizer are visible in the first 10 hour of the training. Note that the proper learning rate influences the noisiness and speed of the learning.

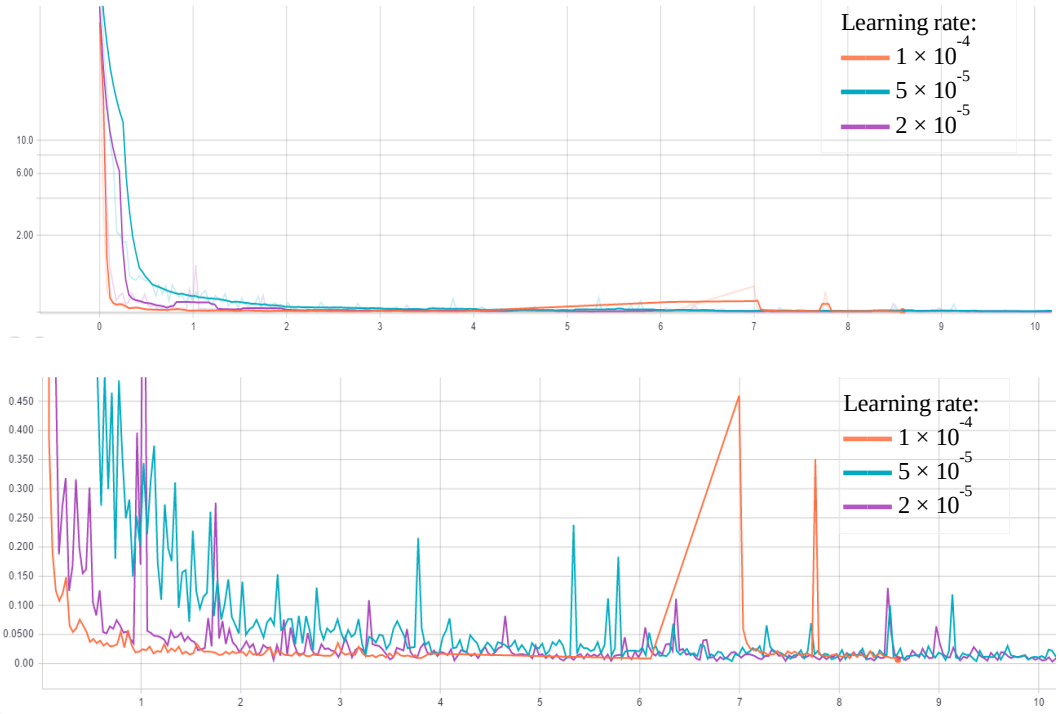

Fig. 4. *The cross-entropy loss with Adam Optimizer at different learning rates*

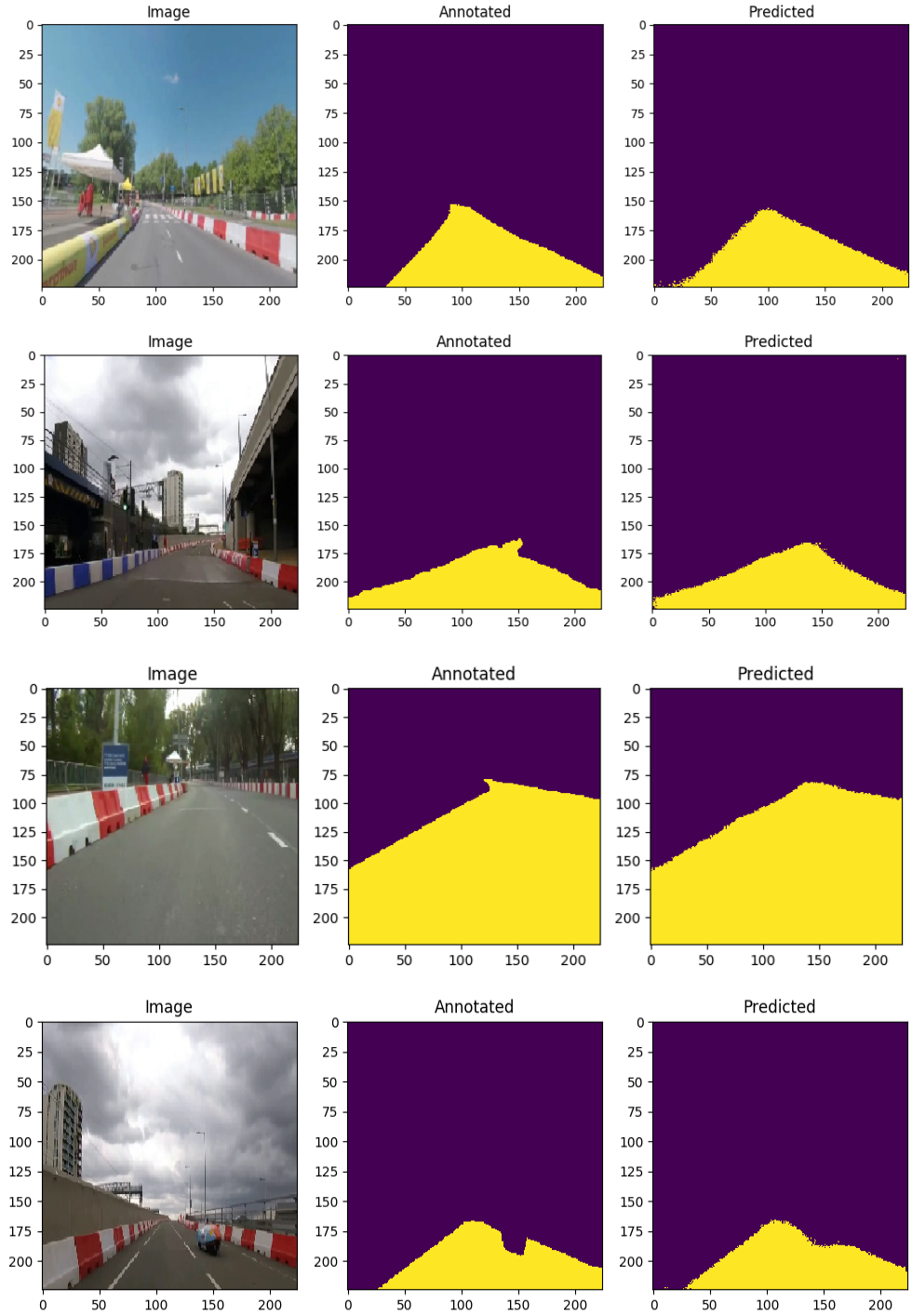

Fig. 5. *Visualisation of the results*

In Figure 5, the first results are visualized with Matplotlib Python library. As it is visible, the network learned the task successfully, but fine-tuning of the results is possible.

As visible, the network generally performs well but there are some uncertainties present at the edges of the road especially when an object e.g. a vehicle is present. A solution for that can be more annotation category, so the network learns more accurately these edges.

5. Possible Future Improvements

As mentioned, the presented research is in the early stage, so several possibilities are available for further development. We will mention the most promising ones:

- GPU implementation;
- Learning rate decay;
- Optimizer;
- More inhomogeneous images;
- Another network design;
- More categories: e.g. road boundary, vehicle category;
- Code diversification.

For the sake of speed GPU implementation of the algorithm would be the most obvious choice. This is also a high-priority request in the real life application, so this will be one of the first improvement what we will make. Of course, leaning speed can be accelerated by quicker learning rate but this may lead to noisy accuracy curve. This can be avoided by continuous learning rate decay. It is also worth investigate in choosing different optimizer or training with more images. For the sake of accuracy more annotation categories and code diversification is the most promising. The software diversification would incorporate a classical CV approach besides the presented NN one, thus the results would be supported from more sources.

To summarize, it is clear the presented neural network approach of road detection is promising and with the mentioned improvements it can be used as a subsystem at the Shell Eco-marathon's challenge.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., et al.: *TensorFlow: A System for Large-Scale Machine Learning*. In: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA, 2016.
2. Badrinarayanan, V., Kendall, A., Cipolla, R.: *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation (SegNet)*. The Computing Research Repository (CoRR), abs/1511.00561, 2015.
3. Karpathy, A., et al.: *Convolutional Neural Networks for Visual Recognition*. Notes accompany the Stanford CS class CS231, 2017.
4. Krizhevsky, A., Sutskever, I., Hinton, G.E.: *Imagenet Classification with Deep Convolutional Neural Networks (AlexNet)*. In: Neural Information Processing Systems Conference (NIPS), 2012.

5. Lee, A.: *Comparing Deep Neural Networks and Traditional Vision Algorithms in Mobile Robotics*. Swarthmore College, 2015.
6. Moravec, H.: *When Will Computer Hardware Match the Human Brain?* In: Journal of Evolution and Technology **1** (1998).
7. Shekkizhar, S.: https://github.com/shekkizh/FCN.tensorflow.
8. Simonyan, K., Zisserman, A.: *Very Deep Convolutional Networks for Large-Scale Image Recognition (VGG)*. In: The Computing Research Repository (CoRR), abs/1409.1556, 2014.
9. Shelhamer, E., Long, J., Darrell, T.: *Fully Convolutional Models for Semantic Segmentation (FCN)*. In: PAMI, 2016.
10. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Rabinovich, A.: *Going Deeper with Convolutions (GoogLeNet)*. In: The Computing Research Repository (CoRR), abs/1409.4842, 2014.
11. Teichmann, M.: https://github.com/MarvinTeichmann/tensorflow-fcn.
12. Tensorflow examples: https://github.com/tensorflow/examples/tutorials/mnist.