HUNGARIAN JOURNAL OF
INDUSTRY AND CHEMISTRY
Vol. 48(1) pp. 3–10 (2020)
hjic.mk.uni-pannon.hu
DOI: 10.33927/hjic-2020-02

HUNGARIAN JOURNAL OF INDUSTRY AND CHEMISTRY
HJIC

IMPROVING THE EFFICIENCY OF NEURAL NETWORKS WITH VIRTUAL TRAINING DATA

JÁNOS HOLLÓSI*[1,2], RUDOLF KRECHT[2], NORBERT MARKÓ[2], AND ÁRON BALLAGI[2,3]

[1]Department of Information Technology, Széchenyi István University, Egyetem tér 1, Győr, 9026, HUNGARY
[2]Research Center of Vehicle Industry, Széchenyi István University, Egyetem tér 1, Győr, 9026, HUNGARY
[3]Department of Automation, Széchenyi István University, Egyetem tér 1, Győr, 9026, HUNGARY

At Széchenyi István University, an autonomous racing car for the Shell Eco-marathon is being developed. One of the main tasks is to create a neural network which segments the road surface, protective barriers and other components of the racing track. The difficulty with this task is that no suitable dataset for special objects, e.g. protective barriers, exists. Only a dataset limited in terms of its size is available, therefore, computer-generated virtual images from a virtual city environment are used to expand this dataset. In this work, the effect of computer-generated virtual images on the efficiency of different neural network architectures is examined. In the training process, real images and computer-generated virtual images are mixed in several ways. Subsequently, three different neural network architectures for road surfaces and the detection of protective barriers are trained. Past experiences determine how to mix datasets and how they can improve efficiency.

Keywords: neural network, virtual training data, autonomous vehicle

1. Introduction

Shell Eco-marathon is a unique international competition held by Royal Dutch Shell Plc. This event challenges university students to design, develop, build and drive the most energy-efficient racing cars. Our University's racing team, the SZEnergy Team, has been a successful participant in the Shell Eco-marathon for over 10 years. Two years ago, Shell introduced the Autonomous Urban-Concept (AUC) challenge, which is a separate competition for self-driving vehicles that participate in the Shell Eco-marathon. Participants in the AUC challenge have to complete five different tasks, e.g. parking in a dedicated parking rectangle, obstacle avoidance on a straight track, drive one lap of the track autonomously, etc.

Our long-term goal is to prepare for the AUC challenge. One of the main tasks is to create an intelligent system, which perceives the environment of our racing car, e.g. other vehicles, the road surface, other components of the racing track, etc. In this paper, only the segmentation of the road surface and of the protective barriers is taken into consideration. An approach based on neural networks will determine the segmentation, because such networks are one of the best tools to solve problems concerning visual information-based detection and segmentation, e.g. image segmentation. Many high-performance neural network architectures are available such as AlexNet by Krizhevsky et al. [1], VGGNet by Simonyan and Zisserman [2], GoogLeNet by Szegedy et al. [3], Fully Convolutional Networks by Shelhamer et al. [4], U-Net by Ronneberger et al. [5], ResNet by He et al. [6] and Pyramid Scene Parsing Network by Zhao et al. [7]. Training neural networks requires a large amount of training data. However, in this case, the number of training samples is insufficient, e.g. no training images of protective barriers are available and the generation and annotation of real world data is labour-intensive and time-consuming. Computer simulation environments will be used to generate training data for this task. Some attempts that apply virtually generated data to train neural networks have been made. Peng et al. [8] demonstrated CAD model-based convolutional neural network training for joint object detection. Tian et al. [9] presented a pipeline to construct virtual scenes and virtual datasets for neural networks. They proved that mixing virtual and real data to train neural networks for joint object detection helps to improve performance. Židek et al. [10] presented a new approach to joint object detection using neural networks trained by virtual model-based datasets. In this paper, an attempt is made to show the effects of computer-generated training data on the learning process of different network architectures.

The paper is structured as follows: in Section 2, the virtual simulation environment that is used for generating training data is described; in Section 3, our neural network architectures are presented; in Section 4, the training process of the networks is outlined; in Section 5, our

*Correspondence: hollosi.janos@sze.hu

results and experiences are shared; finally, in Section 6, our conclusions are stated.

2. Our virtual environment

Our aim is to create highly realistic image sets that depict racing tracks which follow the rulebook of the Shell Eco-marathon Autonomous UrbanConcept. In order to ensure repeatability and simple parameter setup, the creation of complete, textured 3D-models of the racing tracks is advised. These simulated environments can be used to create images with desired weather and lighting conditions by scanning the track environment using a camera moving at a predefined constant speed. The images created using this method can be processed further, e.g. segmentation and clustering of different types of objects such as the road surface, protective barriers and vegetation. Based on the characteristics of the predefined task, the requirements of the simulation environment can be enumerated:

- highly realistic appearance,
- easy use of textures,
- fast workflow,
- characteristics definable by parameters (parametric lights, weather conditions),
- modular environment construction,
- importability of external CAD models.

Unreal Engine 4 [11] is a games engine designed for the fast creation of modular simulated environments by the use of modular relief, vegetation and building elements. In these environments, actors based on external CAD models could be used. Fields of engineering that apply different visual sensors and cameras require very similar computer simulation technologies to the video game industry. Video games need to be highly realistic as well as efficient due to limited computational capacity. The requirements are the same for the simulation of vehicles mounted with cameras. Highly realistic computer simulations reduce the cost and duration of real-life tests and camera calibrations. It is also important to mention that by using technology implemented and/or developed by the video game industry, the support of a vast developer community is available.

Since our goal is to develop image-perceiving solutions for the Shell Eco-marathon Autonomous UrbanConcept challenge, it is important to carefully follow the rules of this competition with regard to the racing tracks. The simulated environments and racing tracks created by Unreal Engine strictly follow the rules defined by the aforementioned rulebook. These rules define that the self-driving vehicles have to compete on racing tracks equipped with protective barriers of a known height painted in alternating red and white segments. It is also defined that every racing track consists of three

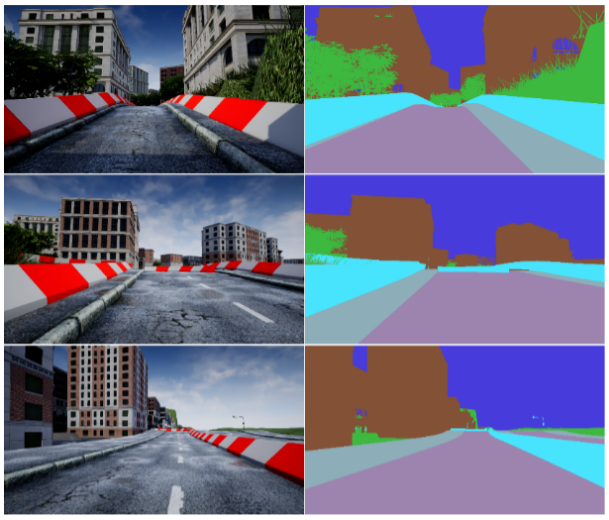

Figure 1: Example images from the training set.

painted line markings, one that is green to denote the starting position, a yellow one to trigger the self-driving mode, and another that is red to mark the finish line. Because the racing tracks and tasks are well defined, it is crucial to create accurate models of the expected environments. Differences between real and simulated environments might lead to further developments in the wrong direction.

Two simulated test environments were created. The first one was based on a readily available city model with streets corresponding to a typical racing track. Barrier elements were added to the roads to ensure that the racing track complies with the requirements outlined in the rulebook. This model includes defects in and textures of the road surface to ensure detection of the road surface is robust. In order to create image sets based on this environment model, a vehicle model equipped with a camera travelled around the racetrack on a pre-defined path. The camera was set to take pictures at pre-defined time intervals. The image set was annotated by using a module called AirSim. AirSim is an open-source, cross-platform simulator simulation platform built on Unreal Engine, but it also has a Unity release. This simulator module consists of a built-in Python-based API (Application Programming Interface) which was developed for image segmentation. By using this API, the necessary realistic and segmented image datasets were created. Some example pairs of images from our virtual dataset are presented in Fig. 1. In order to prepare for all the tasks defined in the rulebook, multiple models of racing tracks were created. All such models are based on the same environment model, which includes vegetation and the sky as shown in Fig. 2. The models of sections of racing track were realized according to the challenges defined in the rulebook. The CAD models representing elements of the racing track were custom-made to comply with the shapes, sizes and colors outlined in the rulebook. The sections of racing track generated can be used to simulate handling

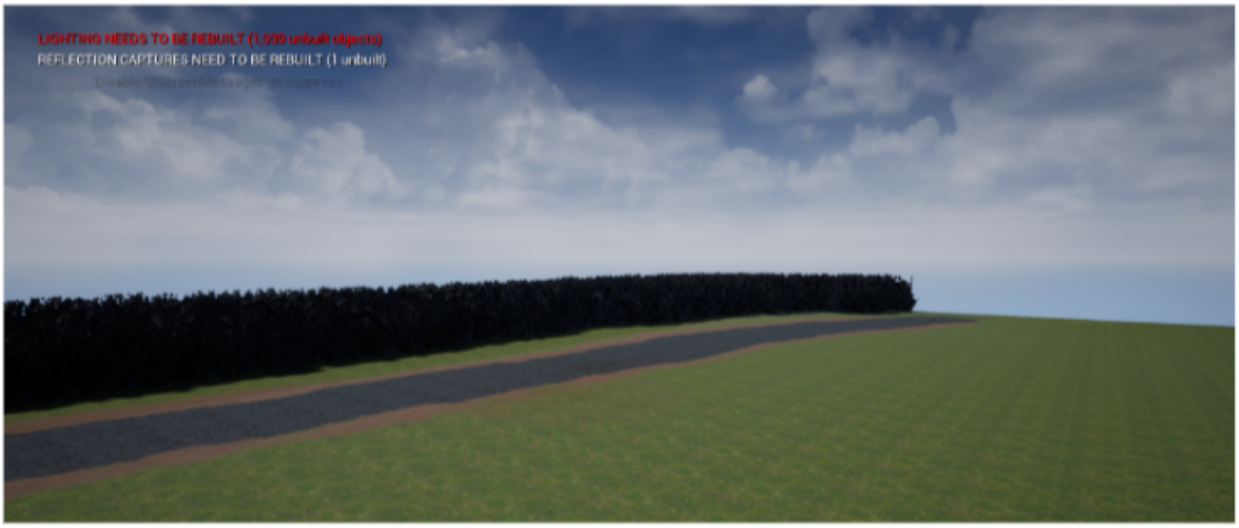

Figure 2: Basic environment of racing tracks.

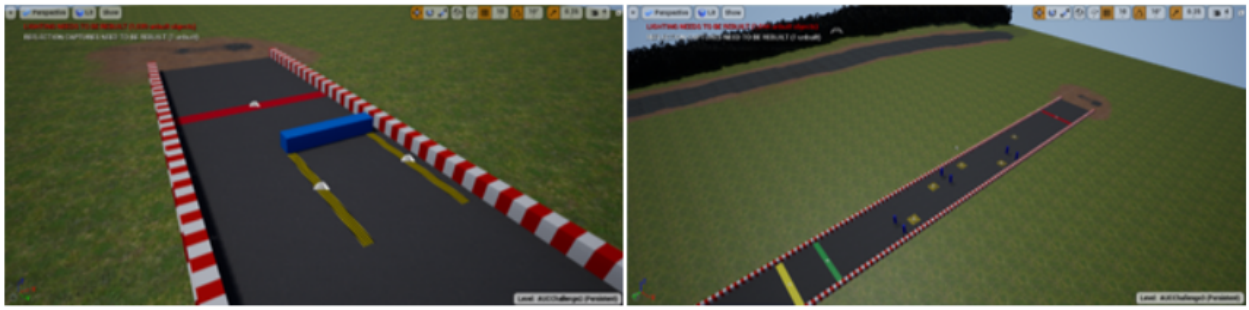

Figure 3: Parking place and slalom course.

(slaloming) and parking tasks. This virtual racing track is shown in Fig. 3. The image sets were created by a moving camera in the environment and segmentation was carried out by changing the textures.

3. Neural network architectures

Three different neural network architectures are implemented in this work: FCN, U-Net and PSPNet. All neural networks are designed for image segmentation, where the size of input images is $256 \times 512 \times 3$, and the size of output ones is $256 \times 512 \times 1$. Every network is trained for the segmentation of the road surface and protective barriers.

3.1 FCN

The Fully Convolutional Network (FCN) [4] architecture is based on fully convolutional layers, where the basic idea is to extend effective classification neural networks to conduct segmentation tasks. Our FCN architectures are shown in Fig. 4.
Let:

$$\gamma = (conv, bn, ReLu) \quad (1)$$

$$b_1 = (\gamma, \gamma, mp) \quad (2)$$

$$b_2 = (\gamma, \gamma, \gamma, mp) \quad (3)$$

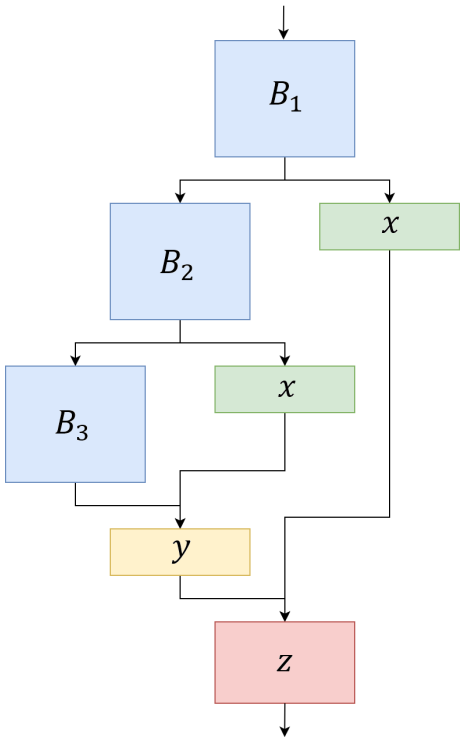

Figure 4: FCN architecture.

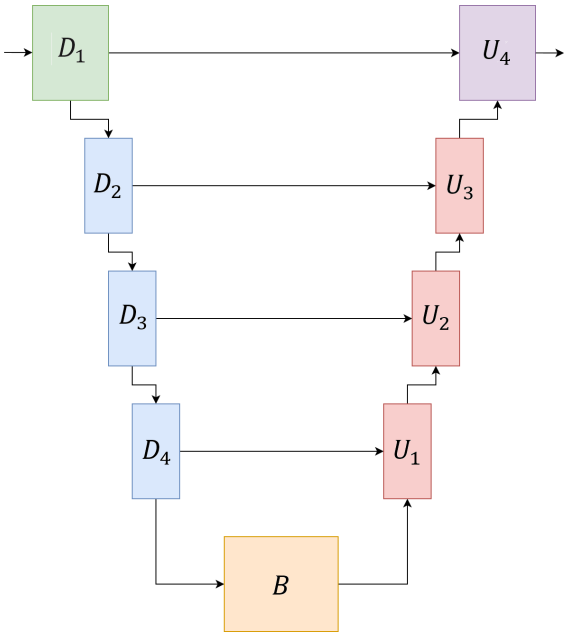

Figure 5: U-Net architecture.

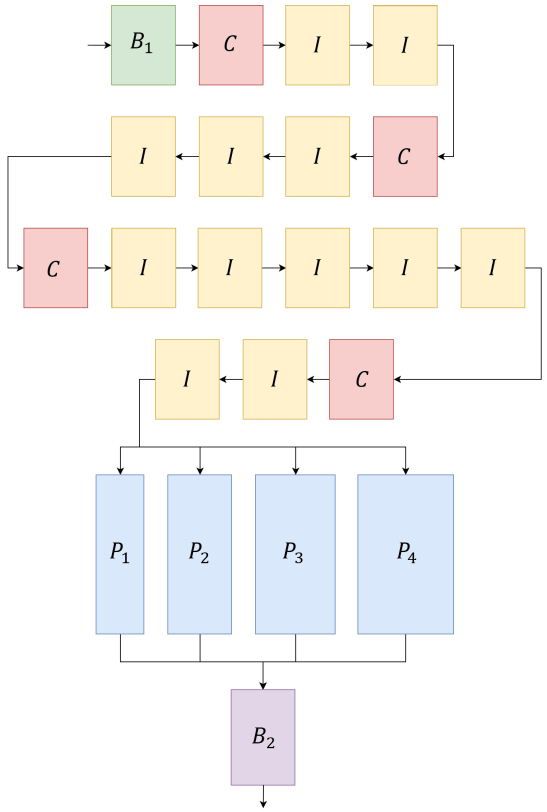

Figure 6: PSP Net architecture.

where $conv$ denotes a convolutional layer, bn represents a batch normalization layer, $ReLu$ stands for a rectified linear activation unit and mp is a max pooling layer. Let:

$$B_1 = (b_1, b_1, b_2) \tag{4}$$
$$B_2 = (b_2) \tag{5}$$
$$B_3 = (b_2, \gamma, \gamma, \gamma) \tag{6}$$
$$x = (conv, bn) \tag{7}$$
$$y = (ReLu, softmax) \tag{8}$$
$$Z = (ReLu, softmax) \tag{9}$$

where $softmax$ denotes a softmax layer. In this implementation, the dimensions of all convolutional layers are 3×3, except for the three fully connected layers in block B_3. The dimensions of these convolutional layers are 7×7. In block B_1, the first two convolutional layers both contain 64 filters, the third and fourth both contain 128 filters, and the last three convolution blocks each contain 256 filters. The convolutional layers in block B_2 contain 512 filters in total. The first three convolutional layers in block B_3 contain 512 filters in total, and the fully connected layers are based on 4096 filters in total.

3.2 U-Net

The U-Net [5] neural network architecture was originally created for biomedical image segmentation. It is based on FCN, where the neural network can be divided into two main blocks, namely the downsampling and upsampling blocks. Our implementations are shown in Fig. 5. Let:

$$D_1 = D_2 = (\gamma, \gamma, maxpooling) \tag{10}$$
$$D_3 = D_4 = B = (\gamma, \gamma, \gamma, maxpooling) \tag{11}$$
$$U_1 = U_2 = U_3 = \left(conv^t, bn, ReLu, \gamma, \gamma\right) \tag{12}$$
$$U_4 = \left(conv^t, bn, ReLu, \gamma, \gamma, conv, softmax\right) \tag{13}$$

where $conv^t$ is a transposed convolution layer. In the U-Net neural network, the dimensions of all convolutions and transposed convolutions are 3×3, and 2×2, respectively. The number of convolutional filters are as follows: each convolutional layer in D_1 consists of 64, in D_2 of 128, in D_3 of 256 and in D_4 as well as B of 512 filters. The upsampling block is very similar. U_1 consists of 512, U_2 of 256, U_3 of 128 and U_4 of 64 filters.

3.3 PSPNet

The Pyramid Scene Parsing Network (PSPNet) [7] was judged to be the best architecture in the ImageNet Scene Parsing Challenge in 2016 [12]. The main building block of the PSPNet is a pyramid pooling module, where the network fuses features under four different pyramid scales. Our PSPNet-based architecture is shown in Fig. 6.

Let:

$$B_1 = (\gamma, \gamma, \gamma, maxpooling) \quad (14)$$
$$C = (\gamma, \gamma, conv, bn) + (conv, bn) \quad (15)$$
$$I = (\gamma, \gamma, conv, bn) \quad (16)$$
$$p = (avg, conv) \quad (17)$$
$$P_1 = (p) \quad (18)$$
$$P_2 = (p, p) \quad (19)$$
$$P_3 = (p, p, p) \quad (20)$$
$$P_4 = (p, p, p, p) \quad (21)$$
$$B_2 = \left(\gamma, dropout, conv, conv^t, softmax\right) \quad (22)$$

where avg denotes an average pooling layer and $dropout$ represents a dropping out unit. In block B_1, the dimensions of all convolutions are 3×3. In blocks C and I, the dimensions of every first & third and every second convolution are 3×3 and 1×1, respectively. In block B_2, the dimensions of the first convolution are 3×3 and the second 1×1. The dimensions of the transposed convolution are 16×16. Each of the first two convolutions in block B_1 consist of 64 filters, and the last one of 128. The first block C and first two I blocks contain 64, 64, 256 filters, respectively, while the second block C and the following three I blocks consist of 128, 128 and 512 filters, respectively. The third block C and the following five I blocks contain 256, 256 and 1024, respectively, and the fourth block C along with the last two I blocks consist of 512, 512 and 2048 filters, respectively.

4. Training with virtual data

An attempt was made to improve the accuracy of neural networks using computer-generated virtual training data that originates from the virtual city environment. Some mixed datasets were compiled which contain real-world images and computer-generated virtual images. The real-world images originate from the Cityscapes Dataset, a large-scale dataset for semantic segmentation [13]. The dataset contains 5000 annotated images with fine annotations created in 50 different cities under various weather conditions. 30 object classes are included, e.g. roads, sidewalks, people, vehicles, traffic lights, terrain, sky, etc. but in this research, only road surface segmentation is examined. The computer-generated images originate from the simulation environment described in Section 2.

For road surface segmentation, five different datasets are created from the Cityscapes Dataset and our collection of virtual images. Table 1 shows how these two collections were mixed. Our goals are to use a minimum amount of data from a real-world dataset, and when the number of virtual images is changed, to observe how the efficiency of the neural networks is affected. Dataset A only contains real-world images, therefore, this is regarded as the basic dataset, while the others were compared to it. Dataset B already contains the same number of virtual images as real-world images. Here, observations of how the introduction of virtual images changes the initial degree of efficiency are sought. Dataset C contains three times more virtual images than Dataset B. If the number of virtual images is much higher than the number of real-world images, the efficiency may be reduced. A future paper of ours will investigate this. In Datasets D and E the number of real-world images was increased. For the segmentation of protective barriers, only virtual training data were used. How efficiently the neural network recognizes real objects, if only trained by virtual data, will now be shown.

Table 1: Number of images in our mixed datasets

Dataset name	Training set		Validation set	
	Virtual	Real-world	Virtual	Real-world
A	0	500	0	125
B	500	500	0	250
C	1500	500	0	500
D	1500	1000	0	625
E	1500	1500	0	750

The effect of increasing the number of real-world images on efficiency was investigated. Adam optimization was used for training with a learning rate of 10^{-4} and a learning rate decay of 5×10^{-4}. As the objective function, categorical crossentropy is used:

$$L\left(y, \hat{y}\right) = -y \times \log\left(\hat{y}\right) \quad (23)$$

and the dice coefficient measured:

$$dc\left(y, \hat{y}\right) = 1 - \frac{2 \times y \times \hat{y} + 1}{y + \hat{y} + 1} \quad (24)$$

where $y \in \{0,\ 1\}$ is the ground truth and $0 \leq \hat{y} \leq 1$ is the result of the neural network.

5. Results

An attempt was made to examine the efficiency of road surface detection, while the composition of the dataset was modified. For examining changes in efficiency, the most useful datasets were A, C and E. Dataset A is the basic dataset, which only contains a small set of real-world images. Dataset C is based on Dataset A, but contains three times as many virtual images as real-world images. Dataset C shows how performance changes, when virtual world images are integrated into a small dataset. In Dataset E, the size of the collection was expanded. This dataset shows how much greater the efficiency of a larger mixed dataset is. Fig. 7 shows the validation accuracy over the training process of road surface detection, while Fig. 8 shows the best dice coefficient values for road surface segmentation. FCN is much simpler than both U-Net and PSPNet neural network architectures.

Hence the efficiency of the FCN on Dataset A is a little less than for the other networks. U-Net and PSPNet are very robust and complex, therefore, mixed datasets do not significantly increase the efficiency of these architectures. However, for simpler networks like FCN, this method improves the efficiency. Fig. 9 shows the perfor-

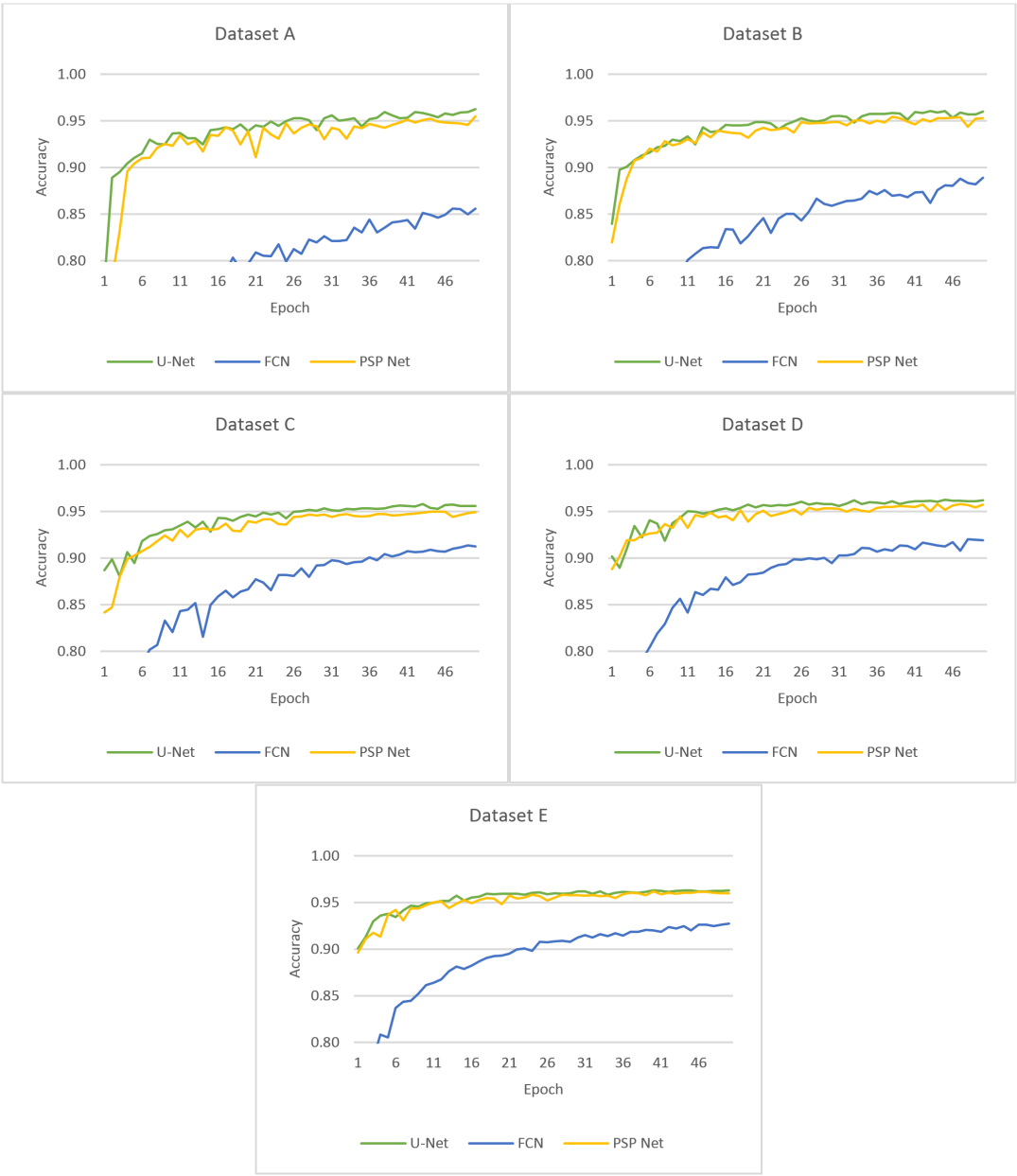

Figure 7: Road surface segmentation performance

mances with regard to the segmentation of protective barriers. Only virtual images were used to train the neural networks that determine the segmentation of protective barriers. This would not have been possible in the case of road surface segmentation, because the road surface is too complex. The texture of the protective barriers is very simple, therefore, it is possible to recognize it from virtual images alone.

It is our intention to use an environment detection system in a low-budget racing car, where the hardware resources available are limited and detection must occur in real time with a high degree of detection accuracy. Therefore, the neural network should be designed to be as simple as possible. If the neural network architecture is too simple, it is more difficult to train for complex recognition tasks. Moreover, the dataset concerning the racing track, protective barriers, etc. is not large. In this case, it is helpful to be able to train simpler neural networks, e.g. FCN, with virtual datasets to achieve higher degrees of efficiency. Experience has shown that the efficiency of road surface detection is improved by using three times as many virtual images, while for protective barrier detection it is sufficient to only use virtual images.

6. Conclusion

This paper presents how to use computer-generated virtual images to train artificial neural networks when the amount of available real-world images is limited. Three different neural network architectures, namely FCN, U-Net and PSPNet, were investigated and these networks trained with mixed datasets. It was shown that virtual im-

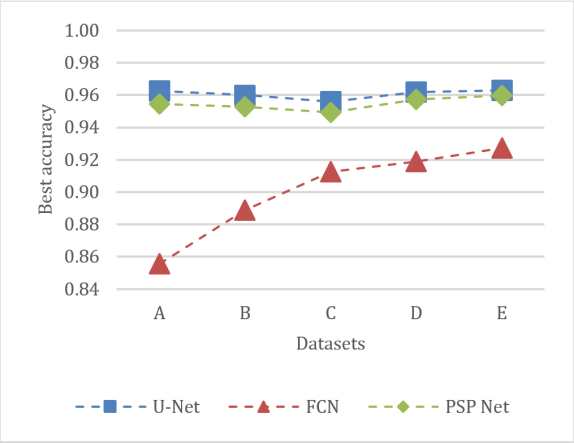

Figure 8: Best accuracy of road segmentation

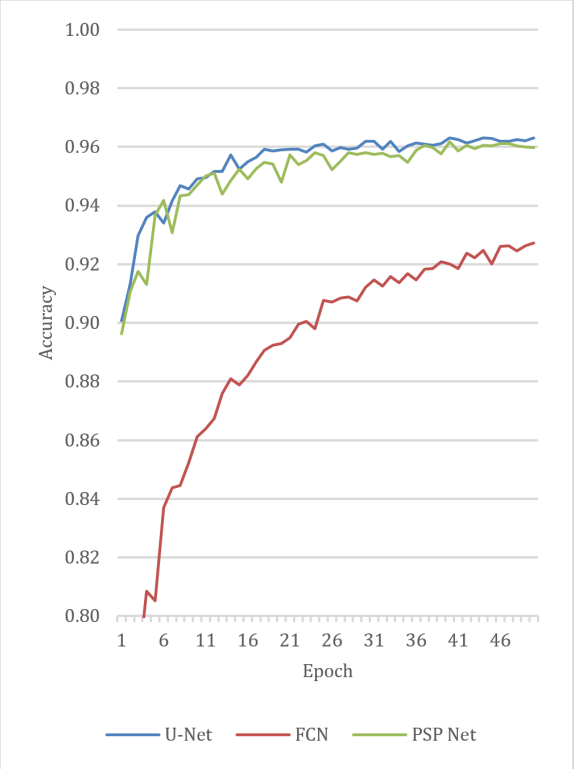

Figure 9: Barrier segmentation performance

ages improve the efficiency of neural networks. Our research demonstrates that when the texture of the objects is simple, e.g. that of protective barriers, it is sufficient to only use virtual image-based training datasets. This work may help us to create an efficient environment detector for the Shell Eco-marathon, where special objects have to be detected in the absence of real-world datasets.

Acknowledgements

The research was carried out as part of the "Autonomous Vehicle Systems Research related to the Autonomous Vehicle Proving Ground of Zalaegerszeg (EFOP-3.6.2-16-2017-00002)" project in the framework of the New Széchenyi Plan. The completion of this project is funded by the European Union and co-financed by the European Social Fund.

REFERENCES

[1] Krizhevsky, G. A.; Sutskever, I.; Hinton, G. E.: ImageNet classification with deep convolutional neural networks, *Commun. ACM*, 2017 **60**(6), 84–90 DOI: 10.1145/3065386

[2] Simonyan, K.; Zisserman, A.: Very deep convolutional networks for large-scale image recognition, 3rd International Conference on Learning Representations, San Diego, USA, 2015

[3] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.: Going deeper with convolutions, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015 DOI: 10.1109/CVPR.2015.7298594

[4] Shelhamer, E.; Long, J.; Darrell, T.: Fully convolutional networks for semantic segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017 **39**(4) 640–651 DOI: 10.1109/CVPR.2015.7298965

[5] Ronneberger, O.; Fischer, P.; Brox, T.: U-Net: convolutional networks for biomedical image segmentation, In: Navab, N.; Hornegger, J.; Wells, W.; Frangi, A. (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015: Lecture Notes in Computer Science, **9351** 234–241, Springer: Cham, Switzerland, 2015 DOI: 10.1007/978-3-319-24574-4_28

[6] He, K.; Zhang, X.; Ren, S.; Sun, J.: Deep residual learning for image recognition, IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 770–778, 2016 DOI: 10.1109/CVPR.2016.90

[7] Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J.: Pyramid Scene Parsing Network, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 6230–6239 2017 DOI: 10.1109/CVPR.2017.660

[8] Peng, X.; Sun, B.; Ali, K.; Saenko, K.: Learning deep object detectors from 3D models, IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 1278–1286, 2015 DOI: 10.1109/ICCV.2015.151

[9] Tian, Y.; Li, X.; Wang, K.; Wang, F.: Training and Testing Object Detectors with Virtual Images, *IEEE/CAA J. Autom. Sin.*, 2018 **5**(2) 539–546 DOI: 10.1109/JAS.2017.7510841

[10] Židek, K.; Lazorík, P.; Pitel, J.; Hošovskı, A.: An automated training of deep learning networks by 3D virtual models for object recognition, *Symmetry*, 2019 **11** 496–511 DOI: 10.3390/sym11040496

[11] Unrealengine.com 2020. Unreal Engine | The Most Powerful Real-Time 3D Creation Platform.

https://www.unrealengine.com/en-US/ [Accessed 14 September 2019]

[12] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.;Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge, *Int. J. Computer Vision*, 2015 **115** 211–252 DOI: 10.1007/s11263-015-0816-y

[13] Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B.: The Cityscapes Dataset for Semantic Urban Scene Understanding, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 3213–3223 2016 DOI: 10.1109/CVPR.2016.350