Sensors & Transducers, Vol. 244, Issue 5, September 2020, pp. 20-27

www.sensorsportal.com

Sensors & Transducers

Published by IFSA Publishing, S. L., 2020
http://www.sensorsportal.com

Capsule Networks for Object Segmentation Using Virtual World Dataset

[1, *] János HOLLÓSI and [2] Áron BALLAGI

[1] Department of Computer Science, Széchenyi University, Egyetem sq. 1., 9026, Hungary
[2] Department of Automation, Széchenyi University, Egyetem sq. 1., 9026, Hungary
[1] Tel.: +36705225635
[*] E-mail: hollosi.janos@sze.hu

Received: 27 July 2020 /Accepted: 31 August 2020 /Published: 30 September 2020

Abstract: The classical convolutional neural networks performance looks exceptionally great when the test dataset are very close to the training dataset. But when it is not possible, the accuracy of neural networks may even be reduced. The capsule networks are trying to solve the problems of the classical neural networks. Capsule networks are a brand new type of artificial neural networks, introduced by Geoffrey Hinton and his research team. In this work we would like to training capsule based neural networks for segmentation tasks, when the training set and test set are very different. For the training we use only computer generated virtual data, and we test our networks on real world data. We created three different capsule based architectures, based on classical neural network architectures, such as U-Net, PSP Net and ResNet. Experiences show how capsule networks are efficient in this special case.

Keywords: Capsule network, CapsNet, Neural network, Object segmentation, Virtual dataset.

1. Introduction

Shell Eco-marathon is a unique international competition for university students to design, develop, build and drive the most energy-efficient race car. Our University's race team, the Szenergy Team has been a successful participant of the Shell Eco-marathon for over 10 years. Two years ago, Shell introduced the Autonomous Urban Concept (AUC) additional competition for self-driving vehicles at the Shell Eco-marathon. AUC competition participants have to complete five different autonomous challenges, like parking on a dedicated parking rectangle, obstacle avoidance on a straight track and so on. One of our main tasks is to create a neural network based intelligent system for this challenge, which perceives the environment of our race car, like the other vehicles, the surface of the road and other special components of the race track. Neural networks are one of the best tools for visual information-based detection and segmentation problems, like image segmentation. Nowadays, many high-performance neural network architectures are available, such as AlexNet by Krizhevsky, *et al.* [1], VGG Net by Simonyan and Zisserman [2], GoogleNet by Szegedy, *et al.* [3], Fully Convolutional Network 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]. However, in this case, we do not have a sufficient number of training samples. For example, we do not have any training image about the protective barriers or the obstacles and it takes a lot of time and energy to generate and annotate real world data. Our idea is to use computer simulation environments to generate training data for this task. In our previously paper [8] we started to working on virtual training

based capsule networks. There have been some attempts that use virtual generated data to train neural networks. Peng, *et al.* [9] demonstrated CAD model-based convolutional neural network training for object detection. Tian, *et al.* [10] presented a pipeline to build virtual scenes and a virtual datasets for neural networks. They proved that mixing virtual and real data to train neural networks for object detection helps to improve the performance. Židek, *et al.* [11] showed a new approach to object detection using neural networks trained by virtual model-based datasets In this paper we continue our previous work, where a brand new and special type of artificial network is used, it is called capsule network [12-13].

2. Capsule Network

The capsule network [12-13] (or CapsNet) is very similar to the classical neural network. The main difference is the basic building block. In neural network we use neurons, but in the capsule network we can find capsules. Table 1 shows the main differences between the classical artificial neurons and the capsules.

The capsule is a group of neurons that perform a lot of internal computation and encapsulate the results of the computations into an n-dimensional vector. This vector is the output of the capsule. The length of this output vector is the probability and the direction of the vector indicates some properties about the entity.

In a capsule based network we use routing-by-agreement, where the output vector of any capsule sent to all higher level capsules. Each capsule output compared with the actual output of the higher level capsules. Where the outputs matches, increase the coupling coefficient between the two capsules.

Let i a capsule and j a higher level capsule. The prediction is calculated as

$$\hat{u}_{(j|i)} = W_{ij} u_i, \tag{1}$$

where W_{ij} is a weighting matrix and u_i is a pose vector for the i^{th} capsule. The coupling coefficients are calculated with a simple softmax function as the following.

$$c_{ij} = \frac{exp(b_{ij})}{\sum_k exp(b_{ik})}, \tag{2}$$

where b_{ij} is the log probability of capsule i being coupled with capsule j.

The total input to capsule j is a weighted sum over the prediction vectors, calculated as the following

$$s_j = \sum_i c_{ij} \hat{u}_{j|i} \tag{3}$$

In capsule networks we use the length of the output vector to represent the probability for the capsule.

Therefore we use a non-linear activation function, it is called squashing function. The squashing function is the next

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \tag{4}$$

We can use the dynamic routing algorithm (Algorithm 1) to update the c_{ij} values. In this case the goal is to maximize the $v_j \hat{u}_{(j|i)}$ value.

Algorithm 1 Routing algorithm [12]

procedure ROUTING($\hat{u}_{j|i}$, r, l)
  for all capsule i in layer l and capsule j in layer$(l+1)$: $b_{ij} \leftarrow 0$
  for r iterations **do**
    for all capsule i in layer l: $c_i \leftarrow softmax(b_i)$
    for all capsule j in layer $(l+1)$: $s_j \leftarrow \sum_i c_{ij} \hat{u}_{i|j}$
    for all capsule j in layer $(l+1)$: $v_j \leftarrow squash(s_j)$
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i} v_j$
  return v_j

Table 1. Differences between capsule and neuron.

		Capsule	**Neuron**
Input		Vector($\boldsymbol{u}_i$)	Scalar (x_i)
Operations	**Affine transform**	$\hat{\boldsymbol{u}}_{j\|i} = \boldsymbol{W}_{ij} \boldsymbol{u}_i$	-
	Weighting	$s_j = \sum_i c_{ij} \hat{\boldsymbol{u}}_{j\|i}$	$a_j = \sum_i w_i x_i + b$
	Sum		
	Non-linear activation	$\boldsymbol{v}_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|^2}$	$h_j = f(a_j)$
Output		Vector ($\boldsymbol{v}_j$)	Scalar (h_j)

3. Virtual World

Our aim is to create highly realistic image sets depicting racetracks which follow the rules defined by the Shell Eco-marathon Autonomous Urban Concept rulebook. In order to ensure repeatability and simple parameter setup it is advised to create complete, textured 3D-models of the racetracks. These simulated environments can be used to create images with desired weather and lighting conditions by scanning the track environment with a camera moving at a previously defined constant speed. The images created using this method can be processed further, including segmentation and the clustering of different object types, such as road surface, barrier elements, vegetation, etc. Fig. 1 shows some example image pair from our virtual dataset. Based on the characteristics of the previously defined task, the requirements of the simulation environment can be enumerated:

- Highly realistic appearance;
- Easy use of textures;
- Fast workflow;
- Characteristics definable by parameters;
- Modular environment construction;
- Importability of external CAD models.

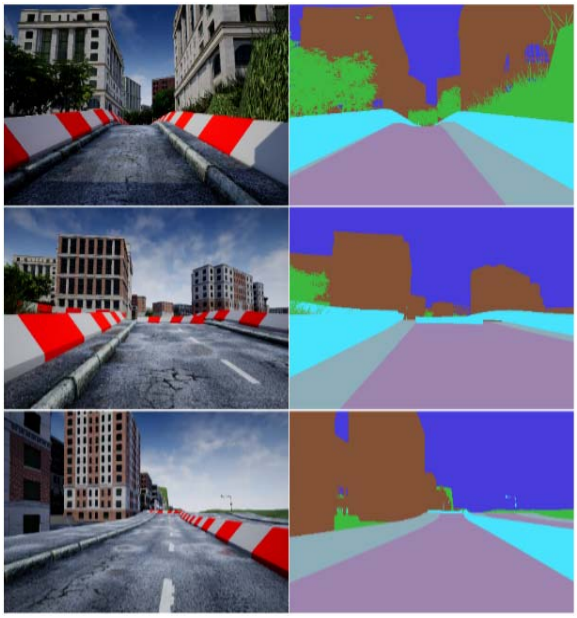

Fig. 1. Example images from the training set.

Unreal Engine 4 [14] is a game engine designed for fast, modular simulated environment creation by the use of modular relief, vegetation and building elements. In these environments, actors based on external CAD models could be used. Engineering fields applying different visual sensors and cameras require very similar computer simulation technologies like computer game industry. Computer games need to be highly realistic and to be efficient at the same time due to limited computational capacity. The requirements are the same in case of 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 computer 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 Urban Concept competition, it is important to carefully follow the rules of this competition regarding the racetracks. The simulated environments, racetracks created with Unreal Engine strictly follow the rules defined by the rulebook mentioned before. The rules define that the self-driving vehicles have to compete on racetracks having side barriers with a known height, having alternating red and white color. It is also defined that every racetrack has three painted marking lines, green one meaning the start position, a yellow one indicating the trigger for self-driving mode, and a red one, which is the finish line. Because the racetracks and the tasks are well defined, it is highly important to create accurate models of the expectable environments. Differences between the real and the simulated environments might lead 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 an expectable racetrack. We added racetrack barrier elements to the roads, and the result is a racetrack which complies with the requirements of the rulebook. This model includes road surface defects and different road surface textures, which makes it possible to make road surface detection 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 built on Unreal Engine, but it also has a Unity release. This simulator module has a built-in Python-based API which was developed for image segmentation. By using this API, we were able to create the necessary realistic and segmented image datasets. In order to prepare for all the tasks defined in the rulebook, we created multiple racetrack models. All the racetrack models are based on the same environment model, which includes vegetation and sky. This environment is shown in Fig. 2. The racetrack section models were realized based on the challenges defined in the rulebook. The CAD models representing track elements were custom created to comply with the rulebook in shape, size and color. The track sections realized can be used for simulating handling (slalom) and parking tasks. Fig. 3 shows this virtual racetrack. The image sets were created by moving a camera in the environment, segmentation was carried out by changing the textures.

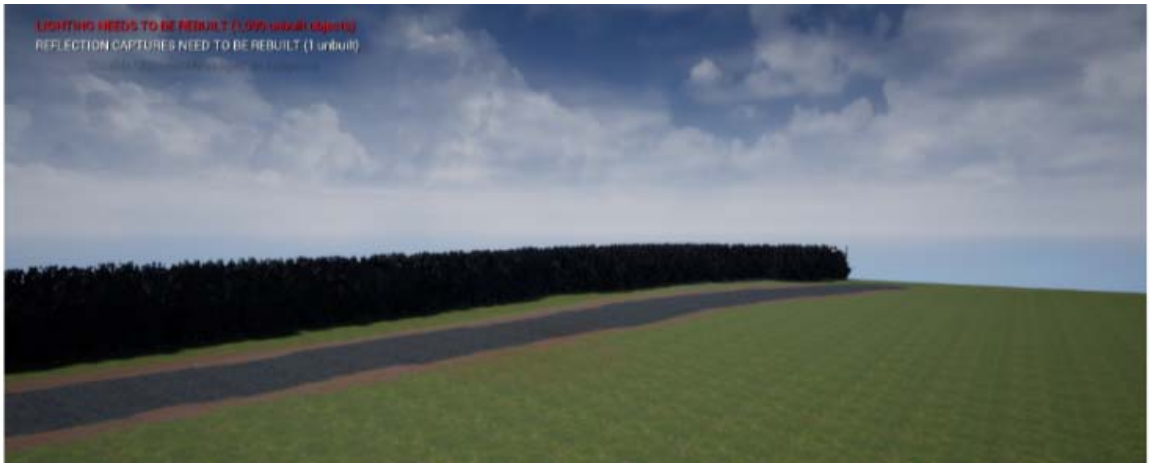

Fig. 2. Basic environment for racetracks.

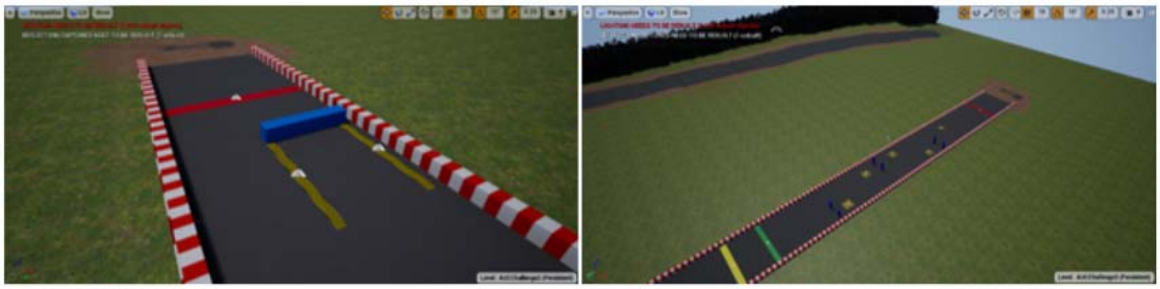

Fig. 3. Parking place and slalom course.

4. Capsule-based Network Architectures

In this work three different capsule based neural network architectures are created. The first capsule network is based on a U-Net [5] architecture, the second contains a pyramid pooling module, based on the PSP Net [7] architecture, and the last one is a residual network based capsule architecture, like ResNet [6].

4.1. U-Net

The U-Net [5] neural network architecture was originally created for biomedical image segmentation. It is based on Fully Convolutional Network, where the network can be divide into two main parts: the downsampling and the upsampling block. Fig. 4 shows our U-net style capsule network architecture.

4.2. ResNet

The ResNet is a very deep neural network, created by He, *et al.* in 2015 [6]. ResNet won the ILSVRC in 2015 with 3.6 % of error rate. The main idea of the Resnet is a residual block. Our ResNet based capsule network architecture consists of two main block: the convolutional block and the identity block.

Fig. 5 and Fig. 6 shows this two blocks, where h is the height, w is the width of the input capsule, c is the number of capsules and a is the number of atoms in every capsule. Fig. 7 shows our ResNet based network.

4.3. PSP Net

The Pyramid Scene Parsing Network [7] is the best architecture on the ImageNet [15] Scene Parsing Challenge in 2016. The main building block of the PSP network is a pyramid pooling module, where the network fuses features under four different pyramid scales. The first phase of our network is built up with the same convolutional block and identity block, which used in ResNet. This is followed by a four stage pyramid block. Fig. 8 shows our PSP Net based capsule network.

5. Results

In this work we created a virtual image based dataset for the protective barrier and road surface segmentation. In our dataset the train samples comes from the virtual city environment, which is presented in 3 and we use only real world images for testing the accuracy of the networks. Our training dataset contains 1572 computer-generated virtual training samples and 131 real world images for validation. The three capsule based networks are trained on this

dataset. Fig. 9 and Fig. 10 shows our results of the U-Net, ResNet and PSP Net based capsule networks. In the training phase Adam optimizer is used with 10^{-1} learning rate and 2×10^{-1} learning decay. The accuracy of the capsule based networks is calculated with dice coefficient

$$dc(y, \hat{y}) = \frac{2 \times y \times \hat{y}}{y + \hat{y}},$$

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

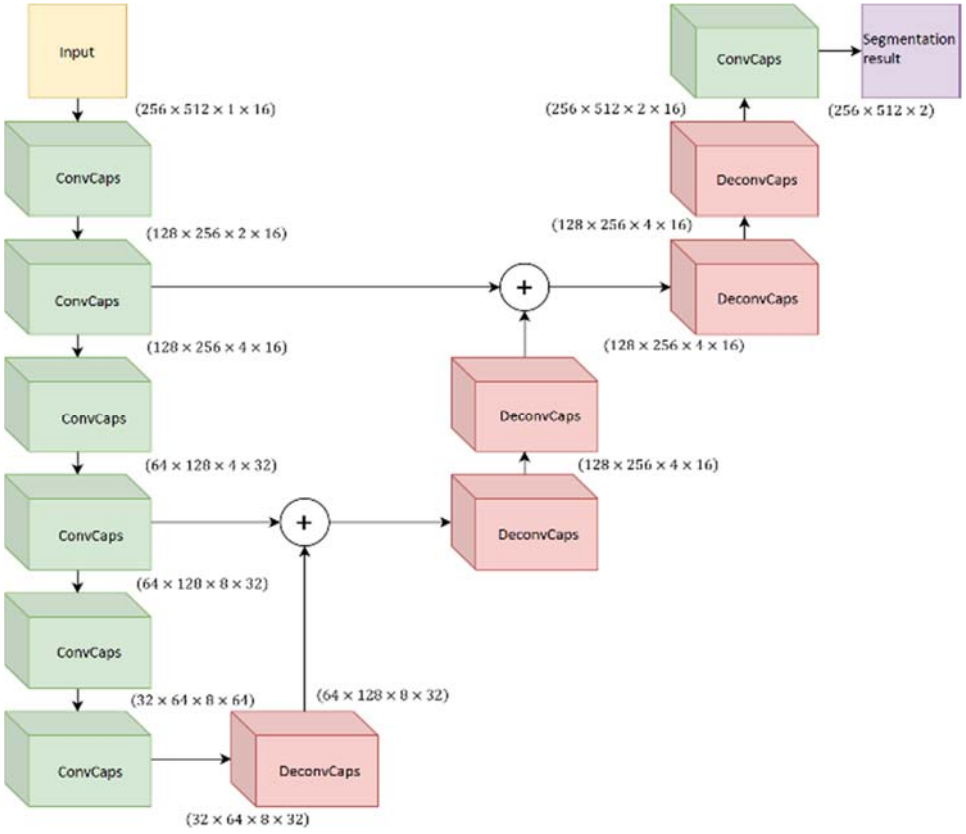

Fig. 4. U-Net architecture.

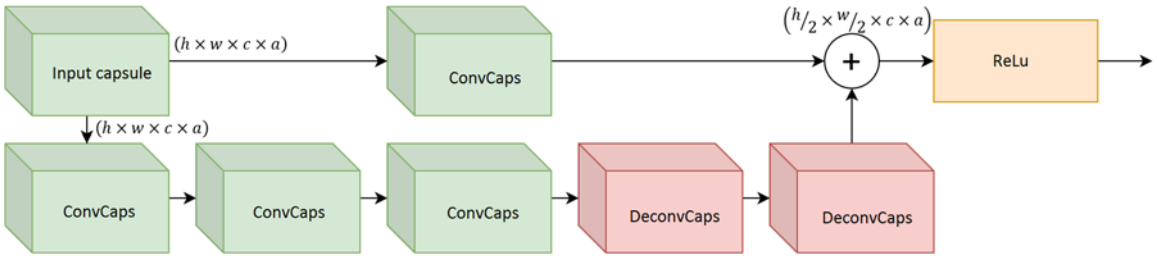

Fig. 5. Capsule based convolutional block.

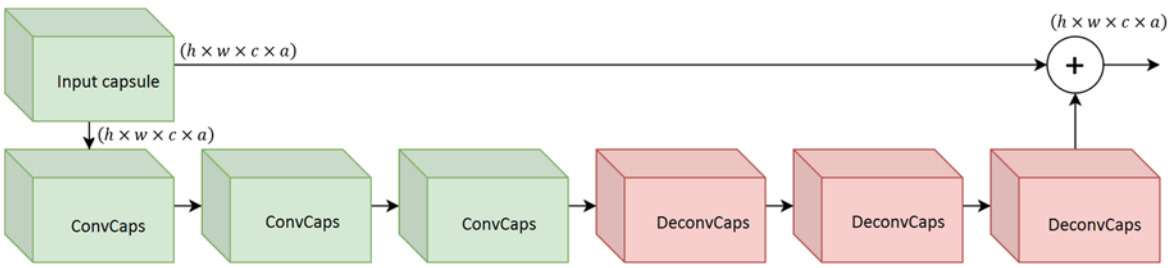

Fig. 6. Capsule based identity block.

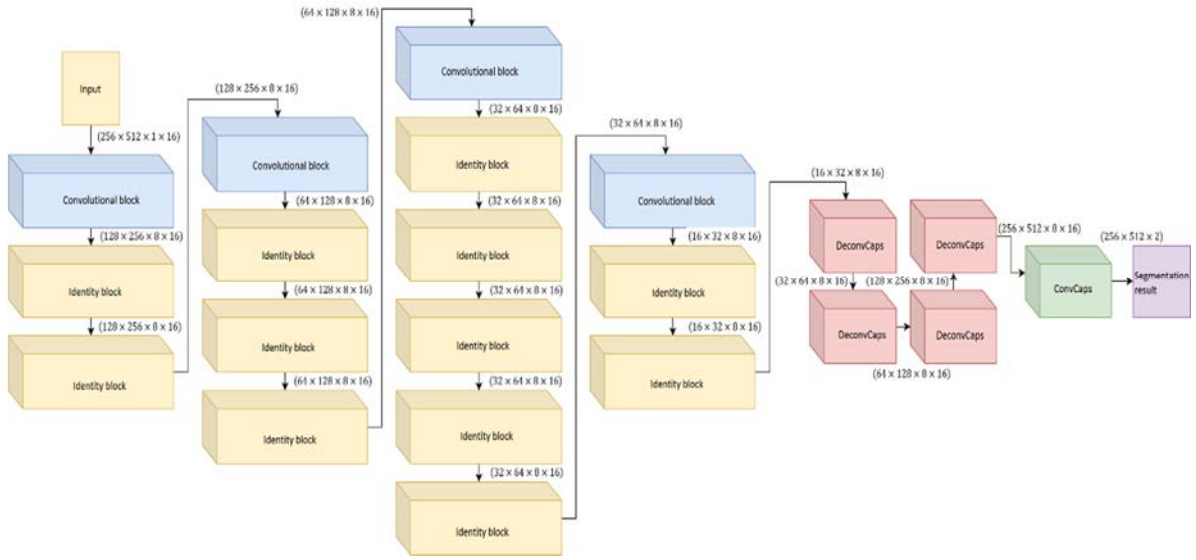

Fig. 7. ResNet Architecture.

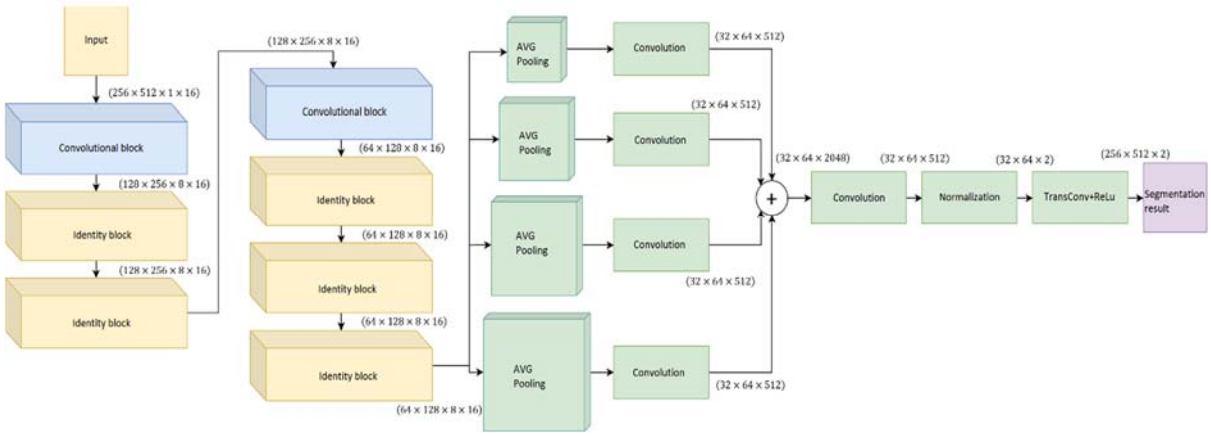

Fig. 8. PSP Net architecture.

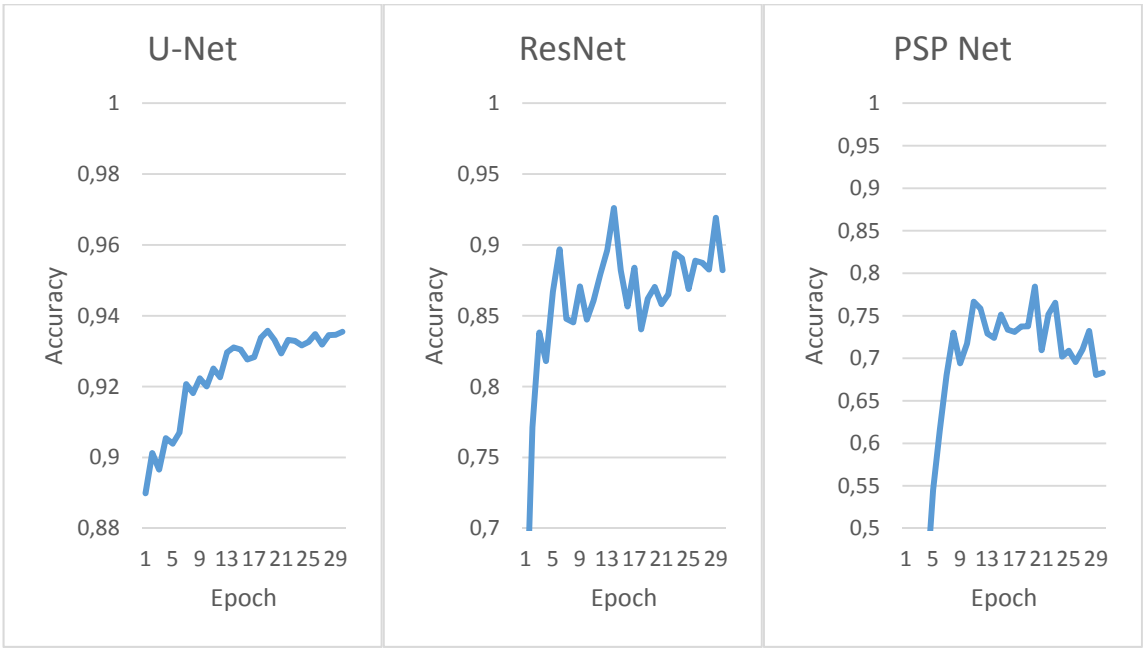

Fig. 9. Accuracy of the capsule based networks.

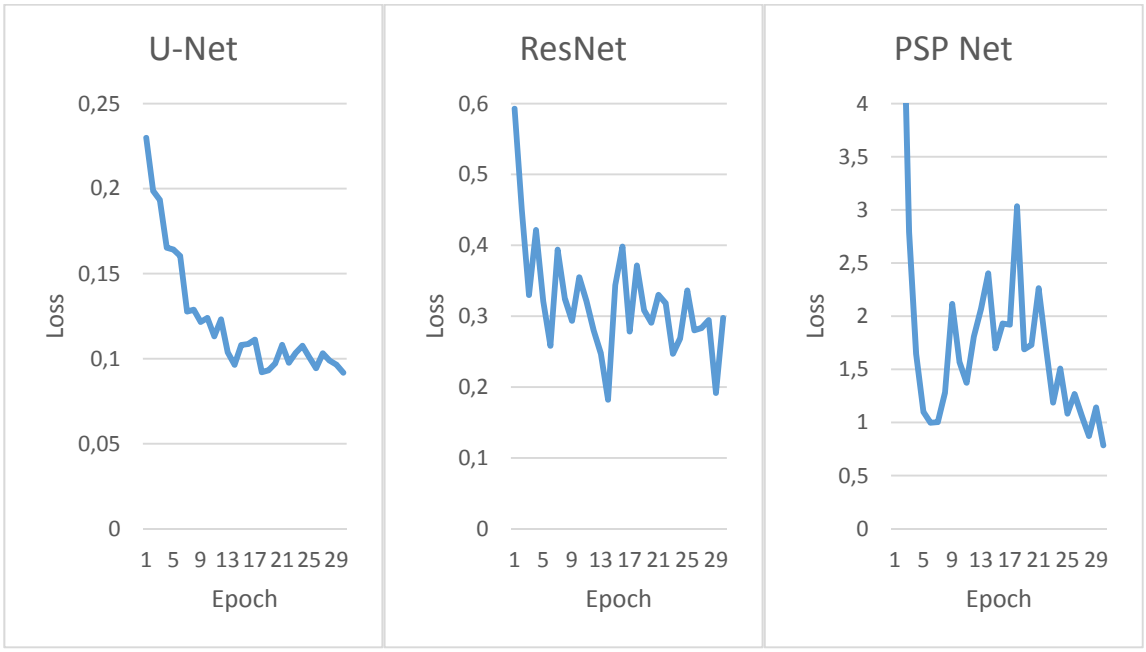

Fig. 10. Loss function of the capsule based networks.

6. Conclusions and Future Work

In this work we training capsule based networks with virtual training data for real world objects segmentation, where our virtual dataset contains computer generated images from a virtual city environment. The results indicate that capsule networks can be used with high reliability in some cases when the size of available dataset is minimal or the training dataset is different from the test dataset. The best results we achieved with the U-Net based capsule network, followed by the residual capsule network, and the last one is the PSP Net. Our experiences shows that in the world of capsule networks, less is more. The U-Net style capsule network architecture is simple but robust. The other two network are more complex, which means lower efficiency in this case. In the future, we would like to achieve higher accuracy in image segmentation tasks with complex capsule based networks. To do this, we would like to examine the effectiveness of the routing algorithm in more detail.

Acknowledgements

Supported by the ÚNKP-19-3-I-SZE-17 New National Excellence Program of the Ministry for Innovation and Technology.

References

[1]. G. A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, in *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS)*, Nevada, USA, Vol. 1, 2012, pp. 1097-1105.

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

[3]. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, Going deeper with convolutions, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015.

[4]. E. Shelhamer, J. Long, T. Darrell, Fully convolutional networks for semantic segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, Issue 4, 2017, pp. 640-651.

[5]. O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, in *Proceedings of the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI'2015)*, Munich, Germany, October 5-9, 2015, pp. 234-241.

[6]. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 770-778.

[7]. H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid Scene Parsing Network, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, USA, 2017.

[8]. J. Hollósi, Á. Ballagi, Training capsule networks with virtual dataset for object segmentation, in *Proceedings of the 2nd International Conference on Advances in Signal Processing and Artificial Intelligence (ASPAI' 2020)*, Berlin, Germany, 2020, pp. 143-147.

[9]. X. Peng, B. Sun, K. Ali, K. Saenko, Learning deep object detectors from 3D models, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015, pp. 1278-1286.

[10]. Y. Tian, X. Li, K. Wang, F. Wang, Training and Testing Object Detectors with Virtual Images, *IEEE/CAA Journal of Automatica Sinica*, Vol. 5, Issue 2, 2018, pp. 539-546.

[11]. K. Židek, P. Lazorík, J. Pitel, A. Hošovský, An automated training of deep learning networks by 3D virtual models for object recognition, *Symmetry*, Vol. 11, 2019, pp. 496-511.
[12]. S. Sabour, N. Frosst, G. E. Hinton, Dynamic routing between capsules, in *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS'17)*, Long Beach, CA, USA, 2017, pp. 3856-3866.
[13]. G. E. Hinton, S. Sabour, N. Frosst, Matrix capsules with EM routing, in *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*, Vancouver, BC, Canada, 2018.
[14]. Unrealengine.com, 2020, Unreal Engine | The Most Powerful Real-Time 3D Creation Platform. [online] Available at: <https://www.unrealengine.com/en-US/> [Accessed 20 July 2020].
[15]. O. Russakovsky, J. Deng, H. Su, *et al*., Imagenet large scale visual recognition challenge, *International Journal of Computer Vision*, Vol. 115, Issue 3, 2015, pp. 211-252.