ELEKTROTEHNIŠKI VESTNIK 88(3): 141-146, 2021
ORIGINAL SCIENTIFIC PAPER

Real-time Portrait Segmentation in TensorFlow

Lejla Hodžić[1], Emir Skejić[2], Damir Demirović[2]

[1] *Bicom doo Tuzla, Bosnia and Herzegovina*
[2] *Faculty of Electrical Engineering, University of Tuzla, Bosnia and Herzegovina*
E-mail: emir.skejic@untz.ba

Abstract. Deep learning is a revolutionizing artificial intelligence, and over the next several decades, it will change the world radically. Many challenging computer vision tasks, such as detection, localization, recognition, and segmentation of objects in an unconstrained environment, are being efficiently addressed by various types of deep neural networks. In the paper semantic segmentation is used to separate a portrait in a video from the background. Semantic segmentation is a task of clustering together parts of an image which belong to the same object class. The aim of the paper is to build four different deep learning models that will be able to segment a portrait from a webcam video in real time and to compare them. Two different deep learning architectures and two different datasets are used. They are both with over 30,000 human portrait images. Our models are trained using TensorFlow which is a novel framework for deep learning and Keras which is a neural network library. Architecture 1 is capable of processing 256×256 RGB images at 12-14 FPS, and Architecture 2 is capable of processing 128×128 RGB images at 15-18 FPS. Our approach achieves a great performance in terms of both the accuracy and efficiency.

Keywords: portrait segmentation, semantic segmentation, TensorFlow, deep learning, Keras

Segmentacija slik v resničnem času v okolju TensorFlow

Globoko učenje močno vpliva na razvoj umetne inteligence. Z globokimi nevronskimi mrežami rešujemo številne zahtevne naloge s področja računalniškega vida, kot so odkrivanje, lokalizacija, razpoznavanje in segmentacija objektov v neomejenem okolju. V tem prispevku uporabljamo semantično segmentacijo za izločitev portreta v videoposnetku. V postopku semantične segmentacije združimo dele slike, ki pripadajo istemu razredu predmeta. Predstavljeni so štirje modeli globokega učenja, ki segmentirajo portret iz videoposnetka s spletne kamere v resničnem času. Uporabili smo dve različni arhitekturi globokega učenja in dva različna nabora podatkov, oba z več kot 30.000 portretnimi slikami. Pri učenju modelov smo uporabili program TensorFlow in knjižnico nevronskih mrež Keras.

1 INTRODUCTION

Computer vision is a science of understanding or manipulating images and videos [1]. It has a lot of applications, including autonomous driving, industrial inspection, and augmented reality. The use of deep learning for computer vision can be categorized into multiple categories: classification, detection, segmentation, and generation, both in images and videos. Deep learning is a collection of techniques from Artificial Neural Network (ANN) which is a branch of machine learning. Artificial neural networks are a computational model that is based on how the brain is believed to work [4]. Artificial neurons are based on the structure of the biological neuron and use mathematical functions with real values to simulate their behavior. Such artificial neurons are called perceptrons. An artificial neuron or perceptron takes several inputs and performs a weighted summation to produce an output. The weight of the perceptron is determined during the training process and is based on the training data. A perceptron can only learn simple functions by learning the weights from examples. The process of learning the weights is called training.

The advancements in computer vision with deep learning have been constructed and improved with time, particularly over one neural network i.e. Convolutional Neural Network (CNN). CNN [6] is a go-to deep learning architecture for computer vision tasks, such as the image segmentation. CNNs have even been extended to the field of video analysis. The CNNs building blocks are filters i.e. kernels. They are used to extract the relevant features from the input using the convolution operation. CNNs have weights, biases and outputs through a nonlinear activation [1]. Regular neural networks take inputs and neurons are fully connected to the next layers. Neurons within the same layer don't share any connection. If regular neural networks are used for images, they will be very large in the size due to a huge number of neurons, resulting in overfitting. An image can be considered as a volume with dimensions of the height, width and depth. A depth is a channel of an image, which is red, blue and green. The CNN neurons are arranged in a volumetric fashion to take advantage of the volume. CNN is a single most important component of any deep learning model for computer vision. It won't be an

Received 2 February 2021
Accepted 15 April 2021

exaggeration to say that it will be impossible for any computer to have vision with no CNN.

Semantic image segmentation of a high accuracy and efficiency using CNNs has been a popular research topic in computer vision. Semantic segmentation [2] is a task of doing a pixel-wise classification. In semantic segmentation, each pixel is classified into one of the predefined sets of classes such that pixels belonging to the same class belong to a unique semantic entity in the image. Creating training data for segmentation tasks is expensive. The semantic segmentation dataset requires specialized software annotators that are very patient and extremely accurate of their work [4]. In fact, the process of labeling with a pixel-level accuracy is perhaps a most time-consuming process among all of the annotation types. For this reason, the number of the semantic segmentation datasets is low and their number of images is limited.

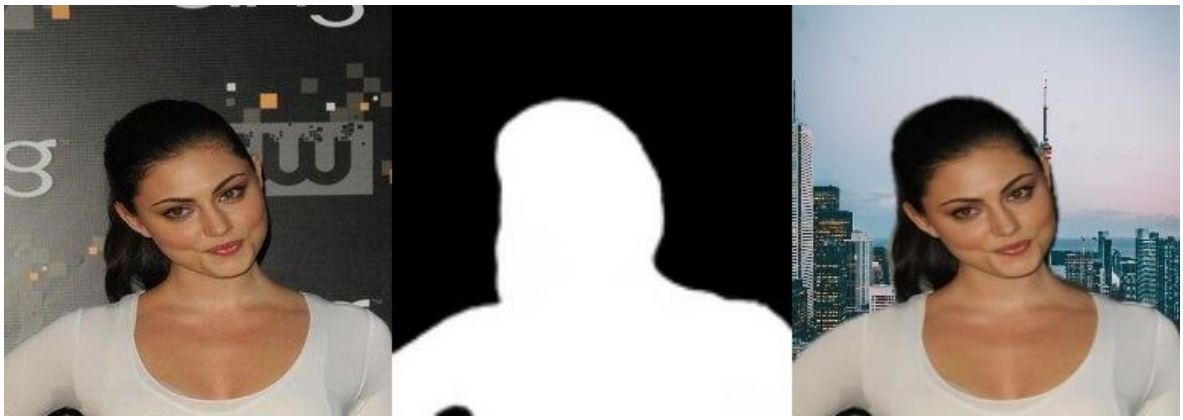
TensorFlow is an end-to-end open source platform for machine learning [5]. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that inspires researchers to promote the state-of-the-art in machine learning and developers to build and deploy machine learning-powered applications. The GPU support exists for specific NVIDIA cards, using the related version of the CUDA toolkit [7]. TensorFlow comes with a strong support for machine learning and deep learning, and the flexible numerical computation core is used across many other scientific domains. The core of TensorFlow is implemented in the C++ programming language, and the main programming language is Python.

TensorFlow uses Keras as a high-level API for its library [3]. It is commonly called tf.keras. Keras is a popular choice of a deep learning library since it is highly integrated into TensorFlow, which is known in production deployments for its reliability. The release of TensorFlow 2.0 has introduced several changes to the framework: from defaulting to eager execution to a complete APIs cleanup [4]. Keras is not a high-level wrapper around a machine learning framework (TensorFlow, CNTK, or Theano); instead, it is an API specification used for defining and training machine learning models. The TensorFlow eager execution is an imperative programming environment that evaluates operations immediately, without building graphs: operations return concrete values instead of constructing a computational graph to run later [5]. TensorFlow 2.0, with its focus on an eager execution, allows the user to design a better-engineered software.

In the paper, TensorFlow 2.0 is used following the Keras API specification. The main programming language is Python 3. Four different deep learning models are built able to segment a portrait from a webcam video in real time. A combination of two architectures and two datasets is used. For both architectures, Convolutional Neural Networks are employed. In Architecture 1, the inputs and outputs are the RGB images of the size of 256×256. It is based on the U-net architecture and uses residual blocks with depth-wise separable convolutions. In Architecture 2, inputs and outputs are the RGB images of the size of 128×128. It is based on the MobileNet-v2 architecture. Training is evaluated on a CPU and GPU – nVIDIA Quadro M1000M.

2 METHODS

Real-time portrait segmentation as a specialized segmentation problem attracts more and more attention, since for web and mobile applications the background editing is very important (e.g. blurring, replacement, etc.) on portrait images and videos (see Figure 1). In this section, two datasets and structure of two architectures used for real-time portrait segmentation are introduced.



a) Original image b) The mask c) Original image after background replacement

Figure 1. Use of portrait segmentation on an image.

2.1 Dataset

The dataset is probably the most critical part of the entire machine learning pipeline [4]. Its quality, structure, and size are the key to the success of deep learning algorithms. A dataset is nothing more than a collection of data. Formally, a dataset can be described as a set of pairs (e_i,l_i), where e_i is the i^{th} example and l_i is its label, with a finite cardinality:

$$\boldsymbol{Dataset} = \{(\boldsymbol{e_i}, \boldsymbol{l_i})\}_{\boldsymbol{i=1}}^{\boldsymbol{k}}$$

A dataset has a finite number of elements. Our machine learning algorithm loops over this dataset several times, trying to understand the data structure, until it solves the task it is asked to address. For training and testing our models, two datasets are used. The first is the "AISegment" [8] portrait dataset. This dataset is currently the largest portrait matting dataset, containing 34,427 images and corresponding matting results. Images are collected from Flickr, Baidu and Taobao and are scaled to 600×800. The corresponding matting files are in the png format, which extracts the alpha map (mask) from the png image before training. Some sample portrait images are shown in Figure 2. To improve the generality of the trained model, several data augmentation methods are used to supplement the original training dataset, leading to better segmentation results. To increase the accuracy of the model in different conditions, cold and warm filters are applied on some images with the help of the Python script. The run-time augmentation methods used in our experiments for both datasets include shift, zoom and horizontal flip and are applied by a Keras data generator and a preprocessing module.

Figure 2. Sample portrait images from Dataset 1.

The second dataset contains 32,711 images and corresponding masks. Images are scaled to 256×256 and masks are in a binary format for two segmentation classes (foreground and background). This dataset is different from the first because it contains the same image five times with different data augmentation methods applied. Those data augmentation methods are: blur, crop and amplified light. Some sample portrait images from this dataset are shown in Figure 3. For both datasets, images are divided into two groups. One is a training set with 80% of the images and the other is a validating set with 20% of the images. The training set is a subset used to train the model and the validation set is a subset used to measure the model performance during training and also to perform a hyper-parameter tuning/search.

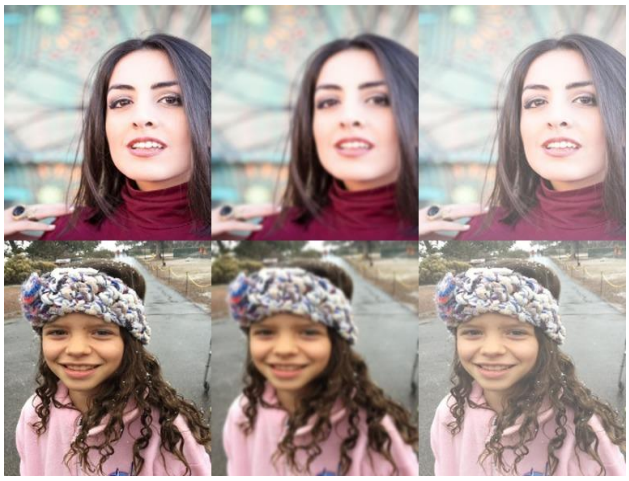


Figure 3. Sample portrait images from Dataset 2.

2.2 Architecture 1

The first architecture used for real-time portrait segmentation is based on the U-Net architecture. The U-Net architecture is designed for semantic segmentation and is based on a fully convolutional network [9]. Our architecture includes two modules, i.e. the encoder and decoder module. RGB images of the size of 256×256 are used as inputs. The encoder module extracts features from a raw RGB image. It is constructed by combining, activated traditional convolution layers and residual blocks with depth-wise separable convolution layers. The depth-wise separable convolution [10] is a variation of the traditional convolution used to improve the efficiency. It performs a depth-wise spatial convolution followed by a point-wise convolution to mix together the resulting output channels. The activation layers use the 'ReLU' activation function, enabling an easier training and better performance. The number of filters, learned by the convolutional layers is 8, 32, 64 and 128. The kernel size is 3×3, the stride is 2×2 reducing the size of the output volume (down-sampling) by 2× as a replacement to max pooling. To reconstruct the spatial information, a decoder module is used. It contains transposed convolution layers for up-sampling the feature maps by 2×. The decoder uses residual blocks too, and keras Add layers instead of concatenating. Our model is implemented using TensorFlow and Keras API. The

Adam optimizer is used to minimize the loss calculated by the binary cross-entropy loss function with the batch size of 64 and weight decay of 1e-3 during training. The initial learning rate is 0.001, and to reduce it when the metric stops to improve, the ReduceLROnPlateau callback is used. To visualize the architectures, TensorBoard is used. Architecture 1 is shown in Figure 4.

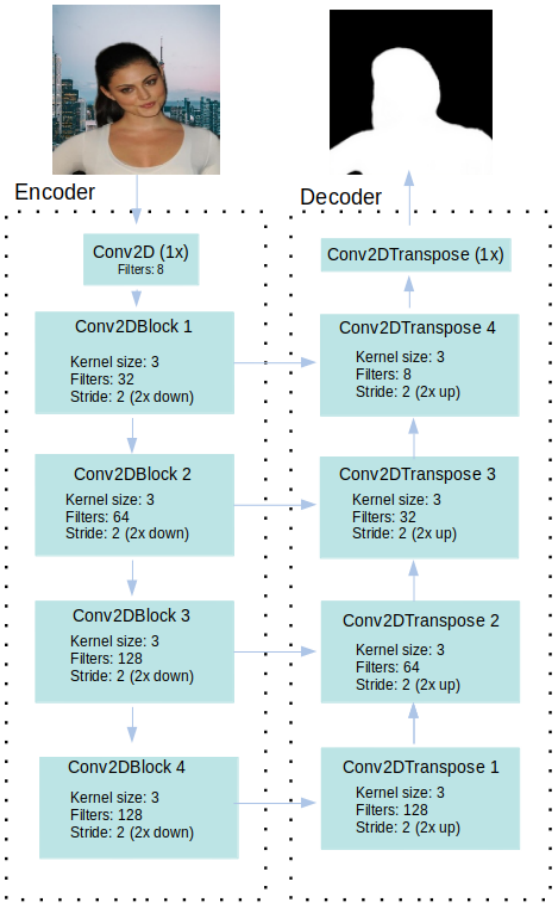


Figure 4. Architecture 1.

2.3 Architecture 2

Architecture 2 used for real-time portrait segmentation is also based on U-Net, but as a backbone for the encoder module the MobileNet-v2 architecture is used [10]. MobileNets are a family of neural network architectures released by Google to be used on machines with a limited computing power. They strive to provide the state-of-the art accuracy, while requiring as little memory and computing power as possible. This makes them a very fast family of networks used for image processing. Our encoder module uses 3×3 depth-wise separable convolutions, width multiplier of 0.5 which proportionally decreases the number of filters in each layer, linear bottlenecks and shortcut connections. The RGB images of the size of 128×128 are used, as inputs and 'imagenet' weights pre-trained on ImageNet. Keeping the U-shape architecture to reconstruct spatial information, a decoder module with UpSampling2D + Conv2D layers for up-sampling the feature maps is used.

The decoder contains batch normalization layers that normalize their inputs, dropout layers with the rate of 0.5 to avoid overfitting and activation layers with a 'ReLU' activation function. Architecture 2 uses concatenate layers in decoder, unlike Architecture 1 which uses Add layers. Our model is implemented using TensorFlow and Keras API. The Adam optimizer is used to minimize the loss calculated by a binary cross-entropy loss function with the batch size of 32 and weight decay of 1e-3 during training. The initial learning rate is 0.001. To reduce it when the metric stops to improve, the ReduceLROnPlateau callback is used. Architecture 2 is shown in Figure 5.

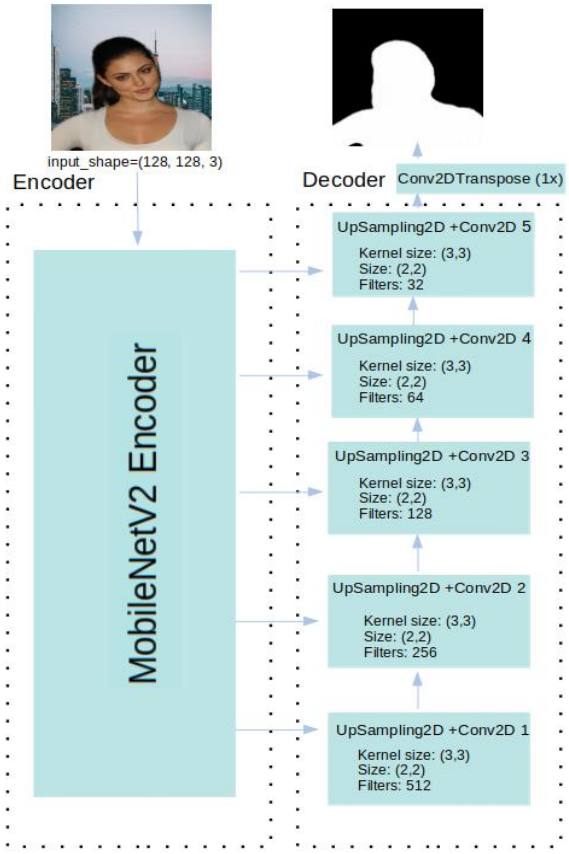


Figure 5. Architecture 2.

3 RESULTS

Training and testing are performed on a single machine running a 64-bit GNU/Linux with seven-core Intel® 6700HQ CPU @ 2.60GHz and GPU NVIDIA Quadro M1000M. Both architectures use the U-Net based architectures to generate sharp segmentation boundaries, depth-wise separable convolutions to gain the running speed, and activation layers using the 'ReLU' activation function to improve the performance and simplify training. Both datasets are large (over 30,000 images), and of a good quality and structure. Figure 6 shows several difficult portrait segmentation results, obtained with a model trained on Dataset 1.

3.1 Accuracy Analysis

The accuracy is a ratio between the number of correct predictions and the number of all predictions made. The accuracy is used to measure the segmentation performance. These metrics are used during the training phase to measure the model performance and monitor how the training proceeds by checking the validation and training accuracy to detect if the model either overfits or underfits the training data. Using Dataset 1, the training accuracy achieved with Architecture 1 is 98.23% and the validation accuracy is 97.71%. For Architecture 2, the achieved training accuracy is 97.17% and validation accuracy is 96.72%. Using Dataset 2, the training accuracy achieved with Architecture 1 is 98.00% and validation accuracy is 96.84%. For Architecture 2 the achieved training accuracy is 97.15% and validation accuracy is 95.97%. Dataset 1 gives slightly better results. The models trained on Architecture 2 achieve higher training accuracy faster, because of using MobileNetV2. This is why in Epoch 1, the accuracy for Architecture 2 is about 94% and for Architecture 1 about 82%. After 10-15 epochs, the accuracy for Architecture 1 increases. The accuracy graph is shown in Figure 7.

Figure 6. Segmentation results of challenging portrait images: a) original image; b) after segmentation on Architecture 1; c) after segmentation on Architecture 2.

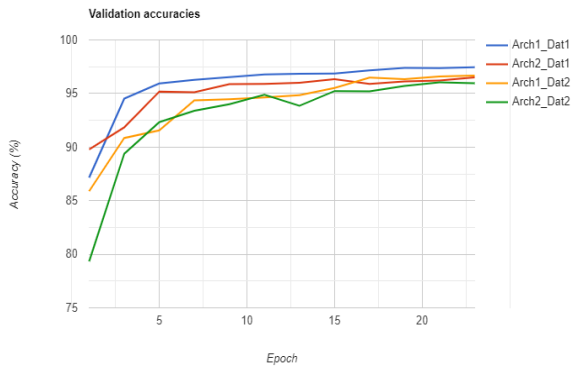


Figure 7. Validation accuracies.

3.2 Loss Analysis

The loss function quantifies how "good" or "bad" a deep learning neural network learns to map a set of inputs to a set of outputs from training data. The loss function has an important job in that it must faithfully distill all aspects of the model down into a single number in such a way that improvements in that number are a sign of a better model. The cross-entropy loss is minimized where smaller values represent a better model than larger values. In the paper a binary cross-entropy loss function is used. Using Dataset 1, the training loss achieved with Architecture 1 is 0.0302 and the validation loss is 0.0451. For Architecture 2, achieved the training loss is 0.0390 and the validation loss is 0.0571. Using Dataset 2, the training loss achieved with Architecture 1 is 0.0401, and the validation loss is 0.0496. For Architecture 2 achieved the training loss is 0.0432 and the validation loss is 0.0585. Following the above the combination of Architecture 1 and Dataset 1 gives the best results. The loss graph is shown in Figure 8.

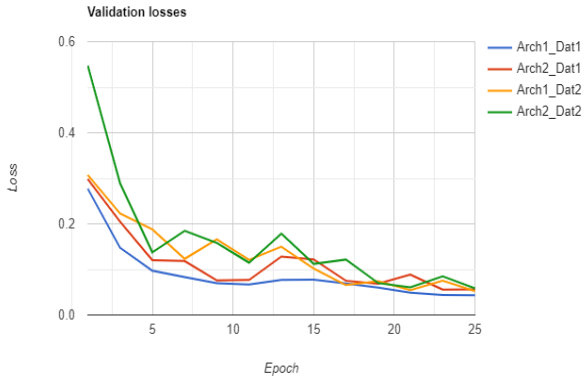


Figure 8. Validation losses.

3.3 Speed Analysis

The efficiency is a very important issue for portrait segmentation in general. For the speed at which images are shown, the models trained on Dataset 1 give slightly better results. Our models are tested using a Python script and also a browser using TensorFlow JS. Regarding the results obtained using the Python script, the model trained on Architecture 1 and Dataset 1 is capable of processing 256×256 RGB images at 12-14 FPS, and the model trained on Architecture 2 and Dataset 1 is capable of processing 128×128 RGB images at 15-18 FPS. The speed comparison for these architectures and datasets is shown in Table 1.

Table 1. Speed comparisons

Architecture	Dataset	Python script	Browser
1	1	12-14 FPS	5-7 FPS
2	1	15-18 FPS	5-8 FPS
1	2	11-12 FPS	5-7 FPS
2	2	14–17 FPS	5-7 FPS

Our method is compared with the representative semantic segmentation method DeepLab [11]. DeepLab uses the PASCAL VOC 2012 dataset consisting of 20 foreground object classes and one background class resulting in 10,582 training images. The DeepLab models segment objects like a person, airplane, train, background on an image, etc. They adopt the simplest form of piecewise training and decoupling the DCNN and CRF training stages, when the unary terms provided by DCNN are fixed. They obtain the highest accuracy using the DeepLab-MSc-CRF-LargeFOV method by segmenting the background on the image. This accuracy is lower than the one obtained for our portrait segmentation and is 93.1%. When it comes to the image processing speed, our method shows a better performance. The DeepLab method operates at 8 FPS, and ours of max 18 FPS. The related comparison graph is shown in Figure 9.

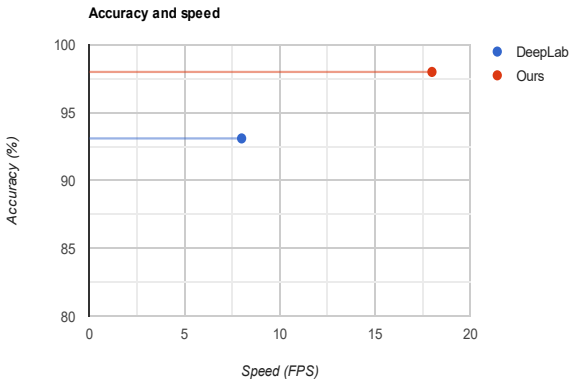


Figure 9. Accuracy and speed comparison graph.

4 CONCLUSION

Real-time portrait segmentation has a significant role in many web applications, such as background replacement or blurring in a video chat or teleconference. The paper presents two architectures and two datasets on which models for segmenting the portraits from a webcam video in real time in a browser are trained. The experimental results demonstrate a high accuracy and good efficiency for both architectures and datasets. Architecture 1 is better in terms of the accuracy (>1%) and loss (~ 0.0088). Architecture 2 is slightly better in terms of the efficiency. Dataset 1 is slightly better than Dataset 2 in all respects, i.e. accuracy, loss and efficiency. Judging from the final portrait segmentation results from a webcam video, the model trained on Architecture 1 gives a more stable output with better boundaries. Therefore, it is recommended to use Architecture 1 and Dataset 1 due to their accuracy and also good results in the measuring the loss and efficiency. However, but both architectures can serve as a useful tool for real-time portrait segmentation in a browser. Compared to the representative semantic segmentation method DeepLab, our method achieves significantly better performance.

In future, the performance gain of the browser using WebAssembly, and minimization of the resource consumption will be investigated.

REFERENCES

[1] Rajalingappaa Shanmugamani, “Deep Learning for Computer Vision: Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras”, 2018.

[2] Swarnendu Ghosh, Nibaran Das, Ishita Das and Ujjwal Maulik, “Understanding Deep Learning Techniques for Image Segmentation”, in ACM Computing Surveys, no. 73, August 2019.

[3] Rowel Atienza, “Advanced Deep Learning with TensorFlow 2 and Keras – Second Edition”, February 2020.

[4] Paolo Galeone, “Hands-On Neural Networks with TensorFlow 2.0”, September 2019.

[5] TensorFlow https://www.tensorflow.org

[6] Analytics Vidhya, “Convolutional Neural Networks (CNN) from Scratch”

[7] CUDA https://docs.nvidia.com/cuda

[8] Matting Human Datasets https://www.kaggle.com/laurentmih/aisegmentcom-matting-human-datasets

[9] J. *Long, E. Shelhamer, T. Darrell, "Fully convolutional networks for semantic segmentation", in IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 640–651 , 2014.*

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, LC. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks”, in IEEE, June 2018.

[11] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected CRFs”, (ICLR International Conference on Learning Representations), 2014

Lejla Hodžić received her B.Sc. degree in Electrical Engineering from the Faculty of Electrical Engineering, University of Tuzla, Bosnia and Herzegovina, in 2018. Currently, she works with the Bicom Systems d.o.o. as a software engineer. Her research interests include digital image processing, computer animation and software engineering.

Emir Skejić received his B.Eng, M.Sc. and Ph.D. degrees in Electrical Engineering from the Faculty of Electrical Engineering, University of Tuzla, Bosnia and Herzegovina, in 2000, 2003 and 2007, respectively. Since 2001, he has been employed with the same faculty, where he is currently an associate professor in the field of Computer and Information Science. His research interests are computer graphics, human-computer interaction, and image processing and analysis.

Damir Demirović received his B.Eng, M.Sc. and Ph.D. degrees in Electrical Engineering from the Faculty of Electrical Engineering, University of Tuzla, Bosnia and Herzegovina, in 2003, 2006 and 2011, respectively. Currently, he is an associate professor at the same faculty. His research interests are in computer science and include pattern recognition, and image processing and analysis.