Suitable ASP U-Net training algorithms for grasping point detection of nontrivial objects

Petr Dolezel[1], Dominik Stursa[1] and Dusan Kopecky[1]

***Abstract*— Robotic manipulation with nontrivial or irregular objects, which provide various types of grasping points, is of both academic and industrial interest. Recently, a powerful data-driven ASP U-Net deep neural network has been proposed to detect feasible grasping points of manipulated objects using RGB data. The ASP U-Net showed the ability to detect feasible grasping points with exceptional accuracy and more than acceptable inference times. So far, the network has been trained using an Adam optimizer only. However, in order to optimally utilize the potential of ASP U-Net, it was necessary to perform a systematic investigation of suitable training algorithms. Therefore, the aim of this contribution was to extend the impact of ASP U-Net by recommending suitable training algorithms and their parameters based on the result of training experiments.**

I. INTRODUCTION

The number of robotic manipulators in industry is steeply growing [1]. Along with this phenomenon, the requirements for higher accuracy, speed, autonomy, multi-purpose applicability, and lower price of the robotic arms are also rising.

One of the most important functionalities of modern robotic arms is the ability to grasp generic objects. Many different ways of robotic object grasping have been investigated. As described by Kumra et al. [2], a robotic grasping system consists of the grasping point detection using a perception system, planning of the movement trajectory, and actual execution of the movement using a robotic arm with a suitable end effector. An example of a robotic grasping system is shown in Fig. 1.

The perception system, as the first module of robotic grasping consecution, is expected to deal with visual processing in order to identify graspable objects and their available grasping points in the scene. Hardware sensors used to scan the area of the scene may be 3-D vision sensors, and conventional RGB or RGB-D cameras [3]. The type of the sensor determines the form of output data (point cloud, RGB-D maps, 2D image, etc.). The important step of the robotic grasping is to detect and locate possible grasping point positions and poses in the provided data.

In the previous authors' work, a powerful data-driven ASP U-Net neural network was proposed to detect feasible grasping points of manipulated objects using RGB data [4]. ASP U-Net is generally applicable for simultaneous grasping point detection considering various types of robotic arm end effectors. In the aforementioned article, the network was extensively examined for the detection of grasping points of the objects scattered in a single layer, using a novel pixel-wise transformation of an RGB image of the scene. Namely, positions and poses of feasible grasping points available in the scene were converted into gradient geometric shapes that effectively encapsulated all the necessary information for a robotic arm to manipulate the objects.

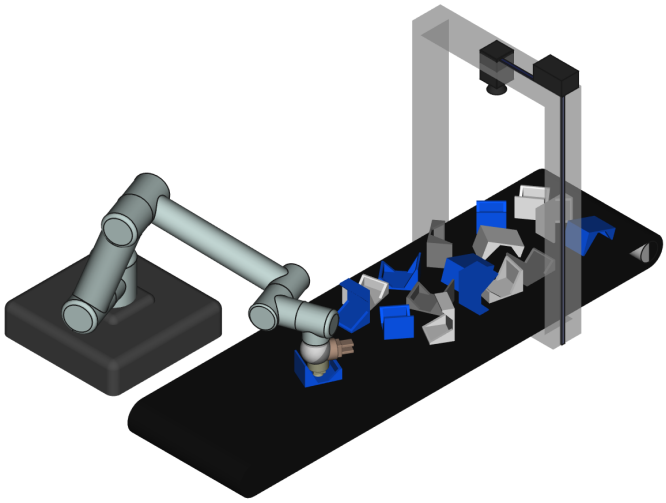

Fig. 1: The demonstration of a robotic grasping system. A conveyor belt moves objects for manipulation, an industrial camera takes an area scan of the scene, and the perception system detects possible grasping points. Lastly, the robotic arm equipped with an end effector places the objects into the target positions.

In line with current trends in image processing, ASP U-Net is based on deep learning and deep neural networks. The network showed the ability to detect feasible grasping points with exceptional accuracy and more than acceptable inference times. As with other data-driven methods based on deep learning, a key step in the design is a training of the neural network. In the above mentioned reference, ASP U-Net was intuitively trained using an Adam optimizer, since it generally provides satisfactory performance in most cases [5], [6]. However, to optimally utilize the potential of ASP U-Net, it would be appropriate to perform a systematic investigation of suitable training algorithms and their parameters. Therefore, the aim of this contribution is to extend the impact of ASP U-Net by recommending a suitable training algorithm and its parameters, based on the result of training experiments.

[1]Authors are with the Faculty of Electrical Engineering and Informatics, University of Pardubice, 532 10 Pardubice, Czech Republic petr.dolezel@upce.cz

II. MATERIALS AND METHODS

ASP U-Net addresses a problem of feasible grasping point estimation in a scene represented by an RGB image. This task is motivated by a common problem of picking various complex objects by a robotic arm and placing them to a required position. The search for a suitable training algorithm for ASP U-Net was performed using the following methods. First, a robotic grasping system used for training and validation data set acquisition was described. Then, ASP U-Net and its functionality was briefly explained. Subsequently, a list of considered training algorithms was defined. Finally, the procedure for identifying the most suitable algorithm was reported.

A. Robotic grasping system

In this contribution, the robotic arm was equipped with two types of end effectors; a parallel gripper and a vacuum cup, as seen in Fig. 1. Additionally, ASP U-Net targeted on objects scattered in a single-layer manner. Such an arrangement doesn't utilize depth information, since all grasping points occur approximately at the same vertical level. Hence, RGB data can be considered as sufficient input source of information.

Furthermore, nontrivial objects with a fixed shape in three color variants (white, blue, gray) were used for the training experiments. The base of the objects was 3 cm high and 4 cm long. The height of the smaller side was 1.3 cm, the height of the larger side was 2.6 cm. The objects provided two kinds of grasping points – edge and plane – for manipulation using either the parallel gripper or the vacuum cup, depending on the object position and pose. The objects with the highlighted possible grasping points are shown in Fig. 2.

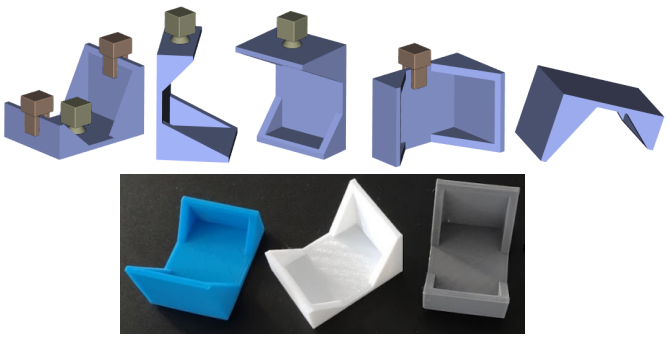

Fig. 2: The shape of the considered nontrivial objects with highlighted grasping points (upper image) and the real object representatives in their color variants (lower image).

Clearly, different spatial arrangements of the objects in the scene can occur, and each individual arrangement differently affects the feasibility of the grasping points for the robotic arm. As extensively shown in the original study [4], the method was designed to successfully deal with the feasibility of the grasping points affected by the mutual influence of surrounding objects. An example is shown in Fig. 3.

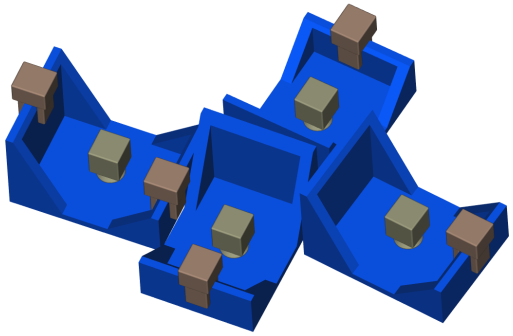

Fig. 3: The feasible grasping points in the scene. Some of the possible grasping points are not feasible due to a neighboring object. These grasping points should not be detected.

B. ASP U-Net for grasping point detection

ASP U-Net was inspired by the U-Net [7], a U-shaped fully convolutional network designed initially for biomedical image segmentation problems. It follows a classical encoder-decoder scheme with a bottleneck. Additionally, it includes skip signals, that allow the network to propagate context information to higher resolution layers. Inspired by SqueezeNet [8] and Squeeze-SegNet [9], conventional convolutional and transposed convolutional layers in ASP U-Net were replaced with special modules called Down sampling module and Up sampling module. According to Ref. [10], these replacements could provide more than 10× parameter reduction while keeping the accuracy. Besides, the attention mechanism, as presented in Ref. [11], was utilized to the decoder part of ASP U-Net. Attention gates filter the features propagated through the skip connections. As a last enhancement, U-Net part of the ASP U-Net architecture was duplicated and mirrored. The mirrored parts were concatenated at the concluding section of the network. Thanks to this enhancement, ASP U-Net is expected to adapt its parameters to process the features relevant to the parallel gripper in the first mirrored branch, and the features relevant to the vacuum cup in the second mirrored branch. The memory requirement of the architecture is 77 MB, and it contains 6.4 million trainable parameters.

ASP U-Net is expected to transform the original RGB image of the scene with the manipulated object into two schematic grayscale images, where the grasping points are highlighted as gradient shapes bearing all the necessary information for a robotic arm. The intended outputs of the network are shown in Fig. 4.

A simplified diagram of the architecture is shown in Fig. 5. Refer to Doležel et al. [4] for the detailed description of the architecture.

C. Training algorithms

Training algorithms are used to change parameters of the neural network (such as weights and biases) to reduce the loss function. Therefore, selection of a particular training algorithm is a crucial step in a neural network design, since

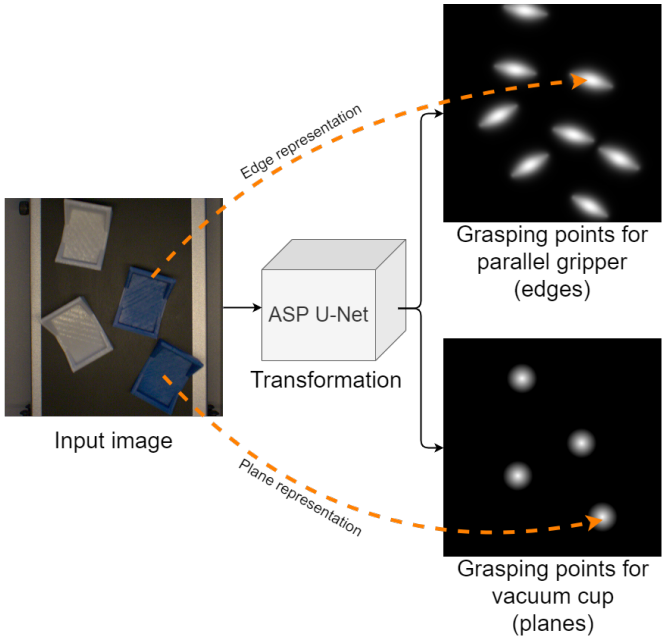

Fig. 4: ASP U-Net transforms the original RGB image of the scene into a pair of schematic images, where the positions of the grasping points are highlighted as gradient shapes. Specifically, the first output from ASP U-Net represents the positions and poses of the grasping points suitable for a parallel gripper (i.e., edges of the objects), and the second output should represent the positions of the grasping points suitable for a vacuum cup (i.e. planes of the objects).

the training algorithm is responsible for providing the most accurate results of the neural network behavior.

Various training algorithms have been researched within the last few decades. According to an extensive literature research, the following algorithms were selected for consideration in this study.

1) Stochastic gradient descent (SGD): SGD is basic but one of the most used algorithms [12]. It is dependent on the first order derivative of a loss function. In its canonical variant, it has only single parameter - learning rate (lr), which determines the step size of each iteration.

2) RMSprop: RMSprop is unpublished optimization algorithm, firstly proposed by Geoff Hinton in his online course. It was developed as the adaptation of Rprop algorithm [13] for batch training. Parameters of the RMSprop include learning rate, ρ (a discounting factor for the history/coming gradient), and ϵ (a small constant affecting numerical stability).

3) Adagrad: This algorithm deals with the learning rate being constant for all parameters and for each iteration [14]. Therefore, it adapts the learning rate individually for each neural network parameter at each iteration, based on previous gradients. Initial learning rate (ilr) has to be set, together with initial accumulator value (iav), which is the starting value for the per-parameter momentum values, and with ϵ.

4) Adadelta: Adadelta is a more robust extension of Adagrad, which tunes learning rates based on a moving window of gradient updates instead of accumulating all past gradients [15]. The parameters include ilr, γ (the decay rate), and ϵ.

5) Adam: Adam training algorithm is a variant of a stochastic gradient descent algorithm, that is based on adaptive estimation of first-order and second-order moments [5]. The parameters to be defined include lr, β_1 (an exponential decay rate for the first order moment estimates), β_2 (an exponential decay rate for the second order moment estimates), and ϵ.

6) Adamax: This algorithm is an extension to the Adam version of gradient descent that generalizes the approach to the infinite norm (max) [5]. Sometimes, it results in a more effective optimization on some problems. This algorithm has the same parameters as Adam.

7) Nadam: Nadam is a variant of the Adam training algorithm with the Nesterov momentum [16]. Like Adamax, this algorithm has also the same parameters as Adam.

D. Procedure for determining the most suitable algorithm

In order to determine the suitable training algorithm, it was necessary to apply all the algorithms comprehensively for ASP U-Net and compare the obtained results systematically. A sufficiently large dataset of scene images and target outputs was needed for this purpose. Additionally, since neural network training is a stochastic process, the training experiments were performed repeatedly to obtain statistically significant results. Last but not least, for selected algorithms it was necessary to set or tune their parameters. All these issues are discussed below.

1) Training dataset: The same dataset, as in the study [4], was used for the training in this contribution. The dataset included 716 original images of a conveyor belt with the manipulated objects taken by a Basler acA2500-14uc industrial RGB camera [17] from a distance of 500 mm. The resulting collection of images included from 0 to 9 objects of three colors with various spatial orientations. Many images contained only parts of the objects. Resolution of the images was 288 × 288 RGB pixels with 8-bit depth.

Each image of the training set had to be annotated in order to be usable for ASP U-Net training. Namely, the grayscale schematic images according to Fig. 4, which defined the positions and orientations of the feasible grasping points, were prepared for each image.

Consequently, the augmentation was used to this dataset using shift operation ($\pm$10 px) and rotate operation ($\pm$10°). Lastly, the dataset was divided into the training (70 %) and validation set (30 %). Some examples from the dataset are depicted in Fig. 6.

2) Training experiments: Two steps of training experiments were performed. Firstly, a promising algorithm had to be chosen, since complete optimization of all parameters of all algorithms was not achievable in time. Therefore, ASP U-Net was trained using each selected training algorithm.

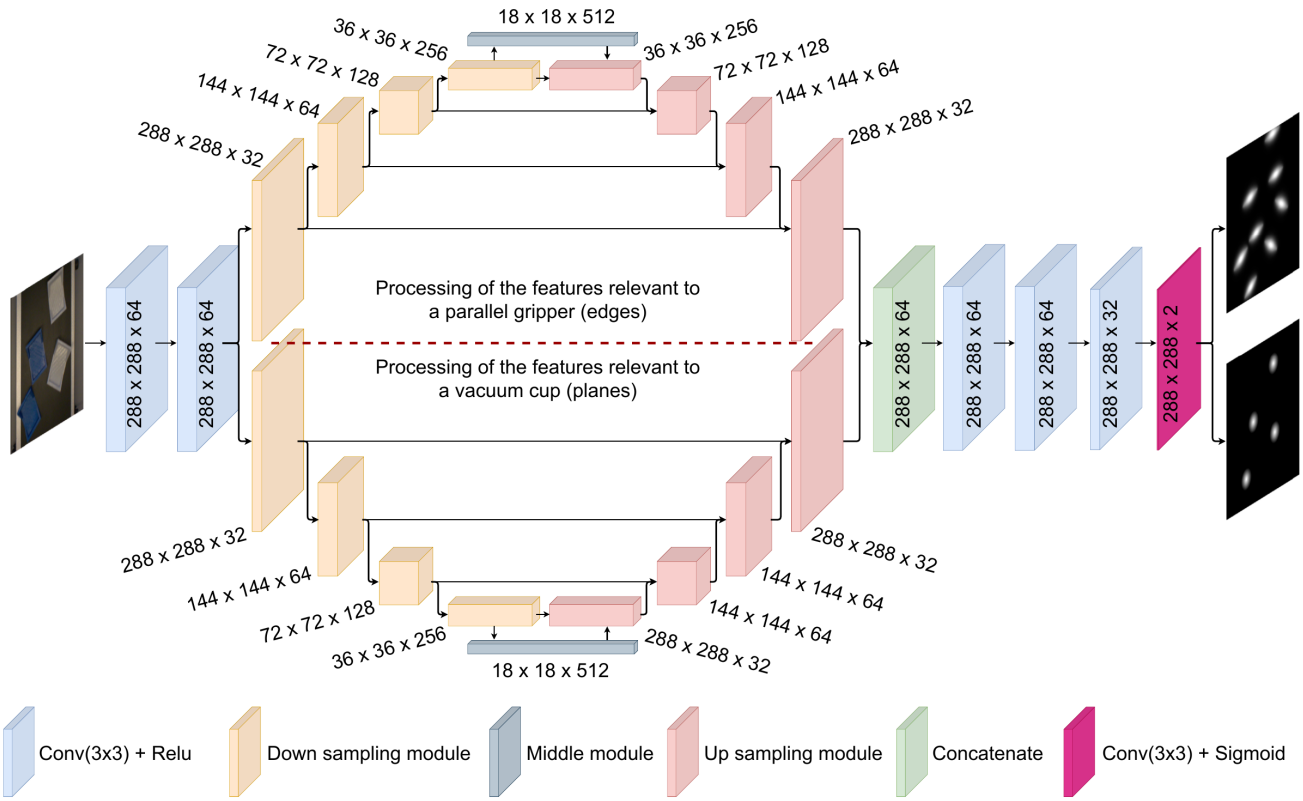

Fig. 5: The ASP U-Net architecture.

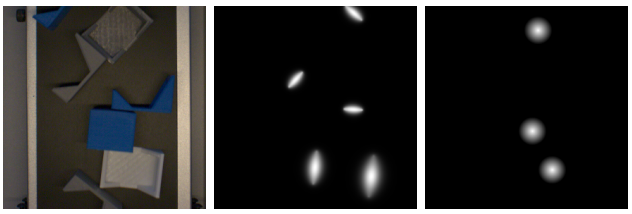

Fig. 6: The example of the images in the training set. Each input image should correspond to two target schematic images. Note that only feasible grasping points are annotated.

Parameters of each algorithm were set based on a compromise between the values obtained from the literature review. Initial weights were set randomly with Gaussian distribution. Binary cross entropy (1) was used as the loss function.

$$\text{loss} = -(y \log(p) + (1 - y) \log(1 - p)), \qquad (1)$$

where y is a binary indicator of correct observation of every pixel in the output image and p is the predicted probability of the observation.

The training experiments were performed twenty times for each training algorithm in order to reduce the stochastic character of the training. All parameters of the training are summarized in Table I.

The second step is to select a promising training algorithm and optimize its parameters. The parameters of the training algorithm with the most suitable and stable performance

TABLE I: Parameters of the training

Input shape	$288 \times 288 \times 3$
Number of experiments	20
Validation split	0.3
Initialization	Normal distribution (mean = 0, std = 0.05)
Number of epochs	100
Crit. for resultant model	Loss function value over validation set
Batch size	4
Parameters for SGD	lr = 0.01
Parameters for RMSprop	lr = 0.001; ρ = 0.9; $\epsilon = 10^{-7}$
Parameters for Adagrad	ilr = 0.001; iav = 0.1; $\epsilon = 10^{-7}$
Parameters for Adadelta	ilr = 0.001; γ = 0.95; $\epsilon = 10^{-7}$
Parameters for Adam	lr = 0.001; β_1 = 0.9; β_2 = 0.999; $\epsilon = 10^{-7}$
Parameters for Adamax	lr = 0.001; β_1 = 0.9; β_2 = 0.999; $\epsilon = 10^{-7}$
Parameters for Nadam	lr = 0.001; β_1 = 0.9; β_2 = 0.999; $\epsilon = 10^{-7}$

in the first step were tuned using another set of training experiments. Specific parameter ranges were not given until the next section because they were dependent on results in the first step.

III. RESULTS

ASP U-Net was trained and validated twenty times according to the procedure addressed in Section II-D. To demonstrate the training results, resulting values of the binary cross entropy loss function evaluated over the validation data at the end of the training sessions were depicted as box graphs in Fig. 7. Central lines in the box graphs, shown in the figure, are medians of loss function; the edges of boxes are 25^{th} and 75^{th} percentiles; and the whiskers extend to

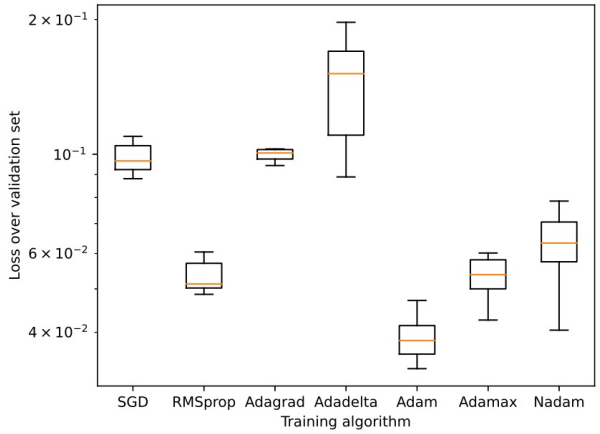

Fig. 7: The box graphs of the binary cross entropy loss function evaluated over the validation data at the end of the training sessions.

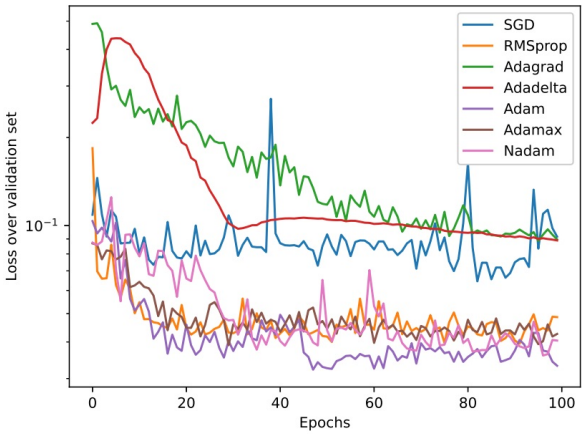

Fig. 8: The learning curves (values of the cost functions related to the epoch number) for the best training sessions.

the most extreme data points (except outliers). Additionally, the best training sessions for each training algorithm were pointed out, and the learning curves (values of the cost functions related to the epoch number) were shown in Fig. 8.

The resulting values of the binary cross entropy loss function indicated that the Adam algorithm was best suited for further investigation. It provided the best overall resulting value of the cost function, while its performance was robust and individual training sessions provided little variance in solutions. In addition, looking at Fig. 8, the Adam algorithm demonstrated a very steep learning curve. Therefore, the Adam algorithm was selected for parameter tuning.

Intuitively, a grid search on log scale was applied to tuning of the parameters. Specifically, β_1 and β_2 remained constant, since it was strongly recommended to use their original values [5], and the rest of the parameters created a grid, where $\epsilon = \{10^{-7}, 10^{-4}, 10^{-1}, 1\}$, lr = $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$.

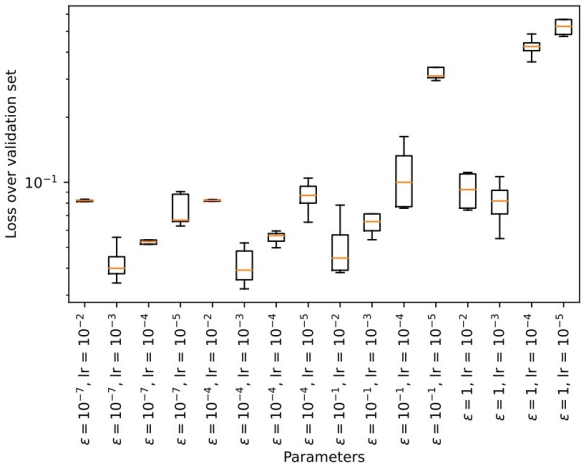

Fig. 9: The box graphs of the binary cross entropy loss function evaluated over the validation data at the end of the training sessions for various parameters of Adam algorithm.

For each combination in this grid, ASP U-Net was trained and validated twenty times analogously to the procedure addressed in Section II-D. To demonstrate the results, the retrieved values of the loss function evaluated over the validation data at the end of the training sessions were again depicted as box graphs in Fig. 9.

Looking at Fig. 9, the pair $\epsilon = 10^{-4}$, lr = 10^{-3} seemed promising. Thus, the neighborhood of this combination of values was examined. Specifically, the grid with the values, where $\epsilon = \{5 \times 10^{-5}, 8 \times 10^{-5}, 3 \times 10^{-4}, 5 \times 10^{-4}\}$, lr = $\{5 \times 10^{-4}, 8 \times 10^{-4}, 3 \times 10^{-3}, 5 \times 10^{-3}\}$ was prepared for investigation. Again, the final values of the cost function over the validation data were shown as box graphs - see Fig. 10.

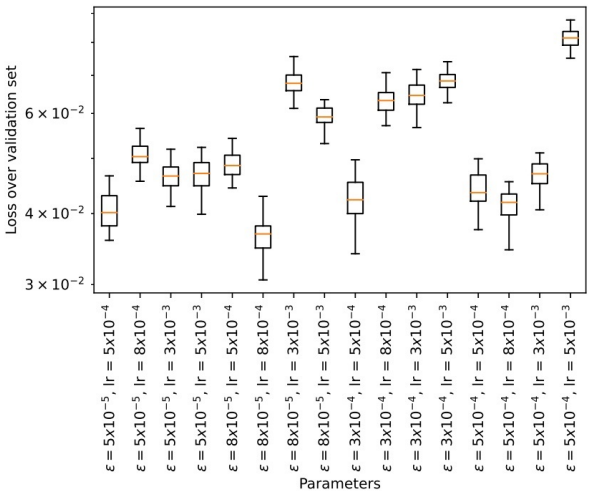

Fig. 10: The box graphs of the binary cross entropy loss function evaluated over the validation data at the end of the training sessions for various parameters of Adam algorithm - fine tuning.

The best overall values were provided by the pair $\epsilon =$

8×10^{-5}, lr = 8×10^{-4}, since this setting provided the best value of the cost function and was a favorable average of all values for this setting as well. However, there is no significant trend in the data presented. Thus, it is possible that the experiment could be extended with a more significant number of training sessions.

IV. CONCLUSIONS

In this contribution, a suitable training algorithm for ASP U-Net was experimentally determined. Moreover, parameters of this algorithm were tuned to provide even better results. Although it is obvious that the search procedure was performed in an engineering way only, the determined algorithm with its parameters can be considered sufficiently robust and efficient for ASP U-Net training.

ACKNOWLEDGMENT

The work was supported from ERDF/ESF "Cooperation in Applied Research between the University of Pardubice and companies, in the Field of Positioning, Detection and Simulation Technology for Transport Systems (PosiTrans)" (No. CZ.02.1.01/0.0/0.0/17_049/0008394).

REFERENCES

[1] I. F. of Robotics, "Robot race: The world's top 10 automated countries," https://ifr.org/ifr-press-releases/news/robot-race-the-worlds-top-10-automated-countries, 2021, 2021-03-08.

[2] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 769–776.

[3] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, 2021.

[4] P. Dolezel, D. Stursa, D. Kopecky, and J. Jecha, "Memory efficient grasping point detection of nontrivial objects," *IEEE Access*, 2021.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[6] E. M. Dogo, O. J. Afolabi, N. I. Nwulu, B. Twala, and C. O. Aigbavboa, "A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks," in *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, 2018, pp. 92–99.

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9351, pp. 234–241, 2015.

[8] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size," 2016.

[9] G. Nanfack, A. Elhassouny, and R. Oulad Haj Thami, "Squeeze-segnet: A new fast deep convolutional neural network for semantic segmentation," vol. 10696, 2018.

[10] N. Beheshti and L. Johnsson, "Squeeze u-net: A memory and energy efficient image segmentation network," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, pp. 1495–1504.

[11] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention u-net: Learning where to look for the pancreas," 2018.

[12] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization Methods for Large-Scale Machine Learning," *SIAM REVIEW*, vol. 60, no. 2, pp. 223–311, JUN 2018.

[13] C. Igel and M. Husken, "Empirical evaluation of the improved Rprop learning algorithms," *NEUROCOMPUTING*, vol. 50, pp. 105–123, JAN 2003.

[14] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *JOURNAL OF MACHINE LEARNING RESEARCH*, vol. 12, pp. 2121–2159, JUL 2011.

[15] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012.

[16] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," no. PART 3, 2013, p. 2176 – 2184.

[17] Basler, "Basler ace," https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca2500-14uc/, 2020, 2021-01-08.