DOI: 10.20535/kpi-sn.2019.1.157259
UDC 004.622:004.021

V.V. Romanuke*
Polish Naval Academy, Gdynia, Poland
*corresponding author: romanukevadimv@gmail.com

AN INFINITELY SCALABLE DATASET OF SINGLE-POLYGON GRAYSCALE IMAGES AS A FAST TEST PLATFORM FOR SEMANTIC IMAGE SEGMENTATION

Background. Every new semantic image segmentation task requires fine-tuning the segmentation network architecture that is very hard to perform on images of high resolution, which may contain many categories and involve huge computational resources. So, the question is whether it is possible to test segmentation network architectures much faster in order to find optimal solutions that could be imparted to real-world semantic image segmentation tasks.
Objective. The goal of the article is to design an infinitely scalable dataset, which could serve as a test platform for semantic image segmentation. The dataset will contain any number of entries of any size required for testing.
Methods. A new artificial dataset is designed for semantic image segmentation. The dataset is of grayscale images with the white background. A polygonal object is randomly placed on the background. The polygon edges are black, whereas the polygon body is transparent. Thus, a dataset image is a set of edges of a convex polygon on the white background. The polygon edge is one pixel thick but the transition between the white background and the polygon black edges includes gray pixels in the vicinity of one-pixel edges. Such a noise is an aftermath of the image file format conversion process. The number of edges of the polygon is randomly generated for every next image. The polygon size and position of its center of mass with respect to image margins are randomized as well.
Results. A toy dataset of any volume and image size from scratch can be generated. Besides, the dataset generator automatically labels pixels to classes "background" and "polygon". The dataset does not need augmentation. Eventually, the dataset is infinitely scalable, and it will serve as a fast test platform for segmentation network architectures.
Conclusions. The considered examples of using the polygonal dataset confirm its appropriateness and capability of networks trained on it to successfully segment stacks of objects. Additionally, a criterion of early stopping is revealed based on empty image segmentation.

Keywords: semantic image segmentation; dataset; polygonal object; transparent background; augmentation; segmentation network architecture; empty image segmentation.

Introduction

Nowadays, semantic image segmentation is the top problem in the field of computer vision. It is a high-level task that paves the way towards complete scene comprehension. Due to processing huge amounts of data, this is a challenge for machine learning also.

Image segmentation is a computer vision task trying to label specific regions of an image, and thus showing what the image contains and where it is located. More specifically, the goal of semantic image segmentation is to label each pixel of an image with a corresponding class or category of what is being imaged [1, 2]. Usually, the image spatial resolution is not allowed to be compressed. So, a semantic segmentation network should classify every pixel in an image, resulting in an image of the same resolution that is segmented by classes or categories.

A few effective approaches towards constructing a neural network architecture for semantic image segmentation exist. They are based on following an encoder/decoder structure [1, 3]. According to the encoder/decoder structure, the spatial resolution of the input is downsampled developing lower-resolution feature mappings, and then the feature representations are upsampled into a full-resolution segmentation map. A common semantic segmentation network consists of a downsampling subnetwork, upsampling subnetwork, and a pixel classification layer [1, 2, 4]. A downsampling subnetwork is stacked of convolutional layers, ReLUs, and max pooling layers [5, 6]. The upsampling is executed using the transposed convolutional layer (also commonly referred to as deconvolutional layer) performing the upsampling and filtering at the same time [7]. An upsampling subnetwork is stacked of deconvolutional layers and ReLUs. The final set of layers performs pixel classifications. These final layers process an input that has the same spatial dimensions (height and width) as the input image. The third dimension being equal to the number of filters in the last deconvolutional

layer is squeezed down to the number of classes that are tasked to be segmented. This is done using a 1-by-1 convolutional layer (in fact, it is a fully-connected layer) whose number of filters is equal to the number of classes. The softmax and pixel classification layers, following the 1-by-1 convolutional layer, categorically label each image pixel.

Surely, it is too early to speak about optimal semantic segmentation network architectures. They are hardly to be deduced purely based on mathematics. Developing optimal architectures takes its time while experience of researchers is accumulated and processed.

The experience has been intensively growing since the late 2000s. As of November 2018, a few tens of image segmentation datasets are available. The most remarkable are COCO, Cityscapes, BSDS500, CamVid, Mapillary Vistas, DUS, PASCAL VOC, MSRCv2, and some others containing labels to each pixel in every image instance [1, 3, 7]. These datasets fit excellently for the corresponding semantic image segmentation tasks but training on them is still expensive. Additionally, augmentation of training data for smaller datasets (like BSDS500 and DUS) is limited. Meanwhile, every new semantic image segmentation task requires fine-tuning the segmentation network architecture that is very hard to perform on images of high resolution, which may contain many categories and involve huge computational resources [2, 7].

Problem statement

So, the question is whether it is possible to test segmentation network architectures much faster in order to find optimal solutions that could be imparted to real-world semantic image segmentation tasks. Such solutions are the number of convolutional layers and their parameters (the size and number of filters) [5, 8], max pooling layers [9], parameters of deconvolutional layers (the size and number of filters), and, probably, dropout layers [10]. For example, a plausible purpose is to research on training data of smaller and simpler images so that real-world tasks could inherit close-to-optimal network architectures from them. At this, whichever a simpler task is, its (simpler) dataset should not need an augmentation [4, 11].

Obviously, such simple datasets can be only artificial. Then, however, the number of their entries is not limited. This is about infinite datasets like EEACL26 [5, 9, 10] or a possible extended version of MNIST where digits would be drawn by a machine simulating inconstancy of human handwriting. In its turn, an infinite dataset for a toy semantic image segmentation must contain primitive objects whose shape and size will vary dramatically. Therefore, the present goal is to design an infinitely scalable dataset, which could serve as a test platform for semantic image segmentation. The dataset will contain any number of entries of any size required for testing. A pattern of how the dataset can be used is going to be eventually exemplified.

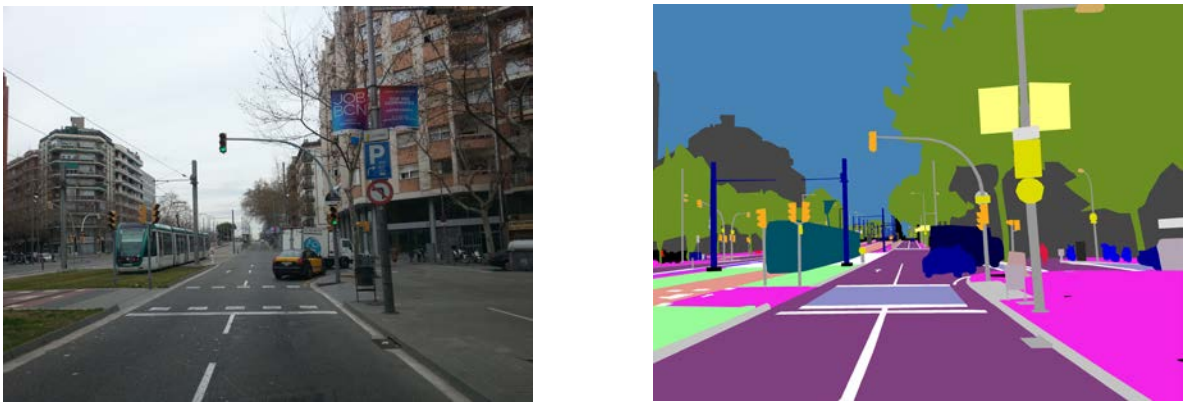
Polygonal objects with transparent bodies to be segmented

A simple image has an object to be segmented on the white background. The object whose color is black must have a geometrical form which could be easily drawn. Such flat objects are polygons. The most primitive form is a triangle. Although polygons having four vertices and more can be concave and self-intersecting, an appropriate choice here is convex polygons [12, 13].

Clearly, the minimal number of polygon vertices is 3. Formally, the maximal number is not limited, so let a polygon be generated of n vertices, where $n \in \overline{\{3, n_{\max}\}}$ by $n_{\max} \in \mathbb{N} \setminus \{2\}$. Number n will be randomly chosen for every new image [14, 15]. Number $n_{\max}$, i. e. a maximal number of edges in a polygon, is specifically selected for a given semantic image segmentation task. Greater number $n_{\max}$ makes this task more complex, and thus a more complex segmentation network architecture may be required (Fig. 1).

As it just has been declared above, the color of the polygon edges is black. What would be the most appropriate color of the polygon interior? It is not necessary to be the same color as edges. If to use discolored images, then it much simplifies the semantic image segmentation task. However, transparency of objects makes this task more complex again because a color interior (say, red or green) would help additionally to identify the color objects on the highly contrasted (white) background [16, 17]. Therefore, images are grayscale (there is no ideal transition between the white background and the polygon black edges). Every image will contain a single polygonal object with transparent body, which has to be segmented.

The minimal image size is 32×32. This is set so for compatibility with the input of convolutional neural networks for other fundamental datasets used in machine learning (like CIFAR-10,



a *b*

Fig. 1. A semantic image segmentation example for more than 10 classes: *a* – an image to be semantically segmented; *b* – a result of segmentation

CIFAR-100, EEACL26, MNIST, etc.). Generally, the image size is $h \times w$, where h is a height, and w is a width (in pixels), which are only limited from below: $h \geqslant 32$, $w \geqslant 32$. Nonetheless, whichever image size would be, the thickness of the polygonal edges will be one pixel. This implies that the segmentation is harder for bigger images because always only a one-pixel border separates the polygon white interior off the white background.

The dataset generator

A procedure of generating a dataset starts with inputting numbers $n_{\min}$ and $n_{\max}$, where $n_{\min} \in \{\overline{3, n_{\max}}\}$. Then an initial number of the polygon vertices is

$$n = \alpha((n_{\max} - n_{\min} + 1) \cdot \theta_1) + n_{\min}, \quad (1)$$

where θ_1 is a value of a random variable uniformly distributed on the open interval $(0; 1)$ by a function $\alpha(\xi)$ returning the integer part of number ξ. Initially, coordinates of the vertices are taken from a vector

$$\mathbf{Y} = (y_i)_{1\times(2n)} = \alpha(\Theta(1, 2n) \cdot \min(\gamma h, \gamma w)) + 1, \quad (2)$$

where function $\Theta(1, 2n)$ returns a pseudorandom $1\times(2n)$ vector [14] whose entries are drawn from the standard uniform distribution on the open interval $(0; 1)$, and γ is a coefficient to scale the polygon with respect to the image size. Theoretically, $\gamma \in (0; 1]$ but $\gamma \in [0.25; 1]$ for practice. The horizontal coordinates are

$$z_i = y_i + \alpha((1 - \gamma) \cdot h\theta_2) \text{ for } i = \overline{1, n} \quad (3)$$

and the vertical coordinates are

$$z_{n+i} = y_{n+i} + \alpha((1 - \gamma) \cdot w\theta_3) \text{ for } i = \overline{1, n}, \quad (4)$$

where values θ_2 and θ_3 are random generated analogously to θ_1. So, the *i*-th vertex is a plane point $[z_i \;\; z_{n+i}]$ for $i = \overline{1, n}$. The stage with vector (2) and coordinates (3) and (4) is repeated until the resulting polygon becomes convex and the coordinates of the same axis are not too close. The latter is controlled by inequalities

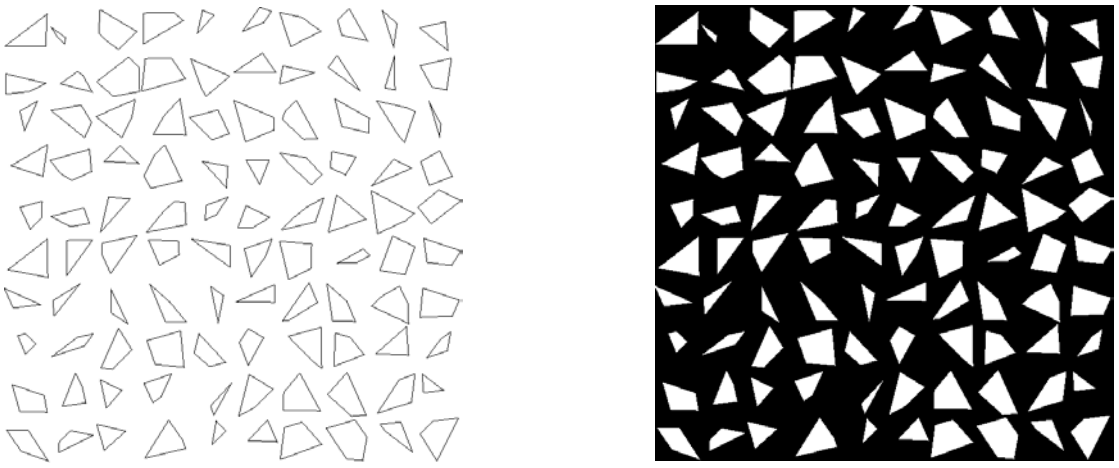
$$\overline{z}_i - \overline{z}_{i-1} \geqslant \lambda \text{ for } i = \overline{2, n} \quad (5)$$

and

$$\overline{z}_{n+k} - \overline{z}_{n+k-1} \geqslant \lambda \text{ for } k = \overline{2, n} \quad (6)$$

for a given positive integer λ, where $\{\overline{z}_i\}_{i=1}^n$ and $\{\overline{z}_{n+i}\}_{i=1}^n$ are respective values $\{z_i\}_{i=1}^n$ and $\{z_{n+i}\}_{i=1}^n$ sorted in ascending order. In practice, a case of $\lambda \in \{\overline{5, 10}\}$ is acceptable. The requirement of inequalities (5) and (6) can be relaxed so that one of them or both may be violated for a single $i \in \{\overline{2, n}\}$ and a single $k \in \{\overline{2, n}\}$. Additionally, one or a few vertices may be deleted in order to obtain the convexity faster [12, 13].

Given a number of entries M, the dataset generator returns M images with polygonal objects and M labeled images (like, e. g., in Fig. 2). Here only two classes are represented: “polygon” and “background”. Later on, the generated dataset is divided into a training set, a validation set, and a testing set (if needed).



a *b*

Fig. 2. An exemplary dataset of 100 grayscale 64×64 images with polygonal objects to be segmented ($n_{\min}=3$ and $n_{\max}=8$): *a* – images of the dataset; *b* – labeled images of the dataset

Optionally, the colors of the background, polygon edges, and polygon interior can be changed. The colors of the background and polygon interior can be different. Note that the images are not ideally of two colors. Some noise is added upon saving an image due to its file format conversion process. A factual transition between the white background and the polygon black edges can be seen in Fig. 3. Besides, due to pixel-wise drawing, the convexity in some places is visibly rough.

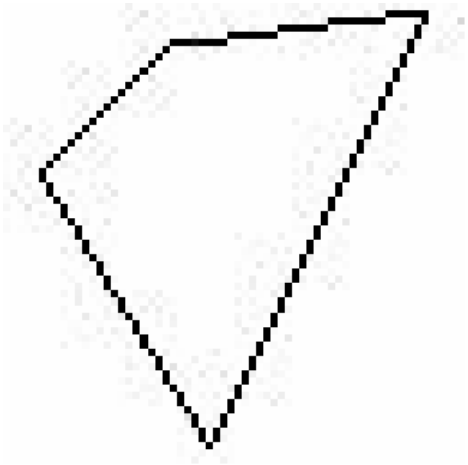


Fig. 3. A grayscale fuzzy transition between the white background and the polygon black edges in an image of the dataset in Fig. 2

A specificity of the dataset generator is that it does not necessarily return a polygon having at least $n_{\min}$ vertices (or edges). While searching for the convex hull by the given coordinates, those vertices who violate the convexity are deleted. This decreases the factual number of polygon edges, especially for small-sized images. As the image size increases, the probability of generating polygons with a greater number of edges increases (for an increased $n_{\min}$). Thus the objects to be segmented can be made a little bit "smoother". However, these objects mainly are triangles, quadrilaterals, and pentagons. Hexagons, heptagons, octagons will be generated much rarer unless numbers $n_{\min}$ and $n_{\max}$ are set greater for 256×256 images or bigger. If to generate vertices by vector (2) until the number of the polygon edges becomes equal to $n_{\min}$, this may significantly linger on it. This is why such an option is not recommended to turn on along with $n_{\min}>6$, although the possibility exists.

Exemplification

Before considering examples of using the polygonal dataset, a common semantic segmentation network architecture is stated as follows. Firstly, the images are all square. So let the input layer have the size $w\times w\times1$. Then a convolutional layer goes executing 3×3 convolutions with unit strides and paddings. A 2×2 max pooling layer downsamples the input by a factor of 2 by setting the stride at 2. A ReLU is inserted in-between the first convolutional and max pooling layers [6].

Let the number of filters in the first convolutional layer be equal to $2w$. And let the second convolutional layer, following the max pooling layer, be of the same parameters. The second convolutional layer is followed by a ReLU also. Further,

a deconvolutional layer upsamples with 4×4 filters whose number is $2w$. The stride here is set at 2.

The deconvolutional layer is followed by 1×1 convolutions with the unit stride and without padding (a fully-connected layer). As we have only two classes, "polygon" and "background", the number of filters here is set at 2. In the end, the softmax and pixel classification layers go (Fig. 4).

Now, four datasets are generated for 32×32, 64×64, 96×96, and 128×128 images. The training and testing sets have an equal number of entries, whereas no validation will be performed during training (eventually, testing is a final validation). Each dataset has three versions: for 400, 2000, and 10000 images.

The network is trained for 100 epochs, which are enough for achieving the top possible performance on a given dataset. The training is executed with using class weighting (Fig. 5). Semantic segmentation quality is evaluated by the common metrics:

```
 1  Image Input                 32x32x1 images with "zerocenter" normalization
 2  Convolution                 64 3x3 convolutions with stride [1  1] and padding [1  1  1  1]
 3  ReLU                        ReLU (by default)
 4  Max Pooling                 2x2 max pooling with stride [2  2] and padding [0  0  0  0]
 5  Convolution                 64 3x3 convolutions with stride [1  1] and padding [1  1  1  1]
 6  ReLU                        ReLU (by default)
 7  Transposed Convolution      64 4x4 transposed convolutions with stride [2  2] and output cropping [1  1]
 8  Convolution                 2 1x1 convolutions with stride [1  1] and padding [0  0  0  0]
 9  Softmax                     softmax
10  Pixel Classification Layer  Cross-entropy loss
```

Fig. 4. The segmentation network architecture in MATLAB for segmenting 32×32 images

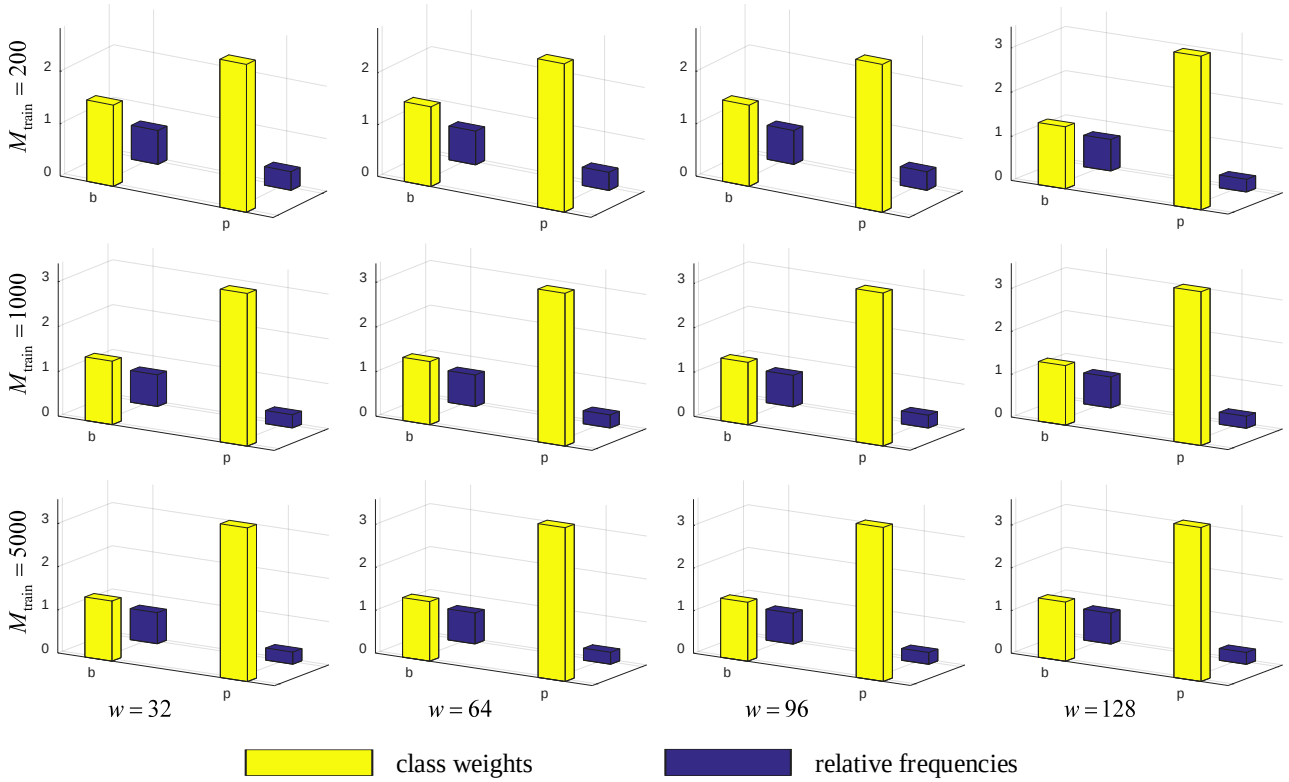


Fig. 5. Relative frequencies of class-labeled pixels and the corresponding class weights, which are multiplicative inverses for the frequencies (the number of class "background" pixels is 1.8206 to 2.6104 times greater than that of class "polygon" pixels)

1. Normalized confusion 2×2 matrix $\mathbf{U}$. The non-diagonal element of this matrix is the count of pixels known to belong to class “polygon”/“background” but predicted to belong to class “background”/“polygon”, divided by the total number of pixels predicted in class “background”/“polygon”.

2. Global accuracy g_{acc}, which is a ratio of correctly classified pixels to total pixels, regardless of class.

3. Mean accuracy m_{acc}, which a ratio of correctly classified pixels in each class to total pixels, averaged over all classes.

4. Mean intersection-over-union (IoU) m_{IoU}, which is the average IoU of all classes.

5. Weighted IoU μ_{IoU}, which is the average IoU of all classes, weighted by the number of pixels in the class.

6. The class accuracy (p_{acc} and b_{acc}, respectively for “polygon” and “background”), which is a ratio of correctly classified pixels in each class to the total number of pixels belonging to that class according to the ground truth. Obviously,

$$m_{\text{acc}} = \frac{p_{\text{acc}} + b_{\text{acc}}}{2}.$$

7. The class IoU (p_{IoU} and b_{IoU}, respectively for “polygon” and “background”), which is a ratio of correctly classified pixels to the total number of ground truth and predicted pixels in that class. Obviously,

$$m_{\text{IoU}} = \frac{p_{\text{IoU}} + b_{\text{IoU}}}{2}.$$

Fig. 6 shows an evolution of matrix

$$\mathbf{U} = \begin{pmatrix} u_{\text{pp}} & u_{\text{pb}} \\ u_{\text{bp}} & u_{\text{bb}} \end{pmatrix}$$

for the increasing image size and the increasing number M_{train} of entries in the training set. The classes are confused badly enough, so the global accuracy (Fig. 7) and mean accuracy (Fig. 8) are far from the acceptable ones. As the image size increases, these accuracies drop. The same happens to the mean IoU (Fig. 9) and weighted IoU (Fig. 10), although they increase as the volume of the training set increases. Tendencies of the class accuracies (Fig. 11) and class IoUs (Fig. 12) differ. However, accuracy and IoU for class “background” resemble each other.

It is no wonder that networks with a bigger input size perform poorer. Although numbers of filters are increased, bigger images may require inserting additional convolutions and deconvolutions along with max pooling layers. Indeed, after insert-

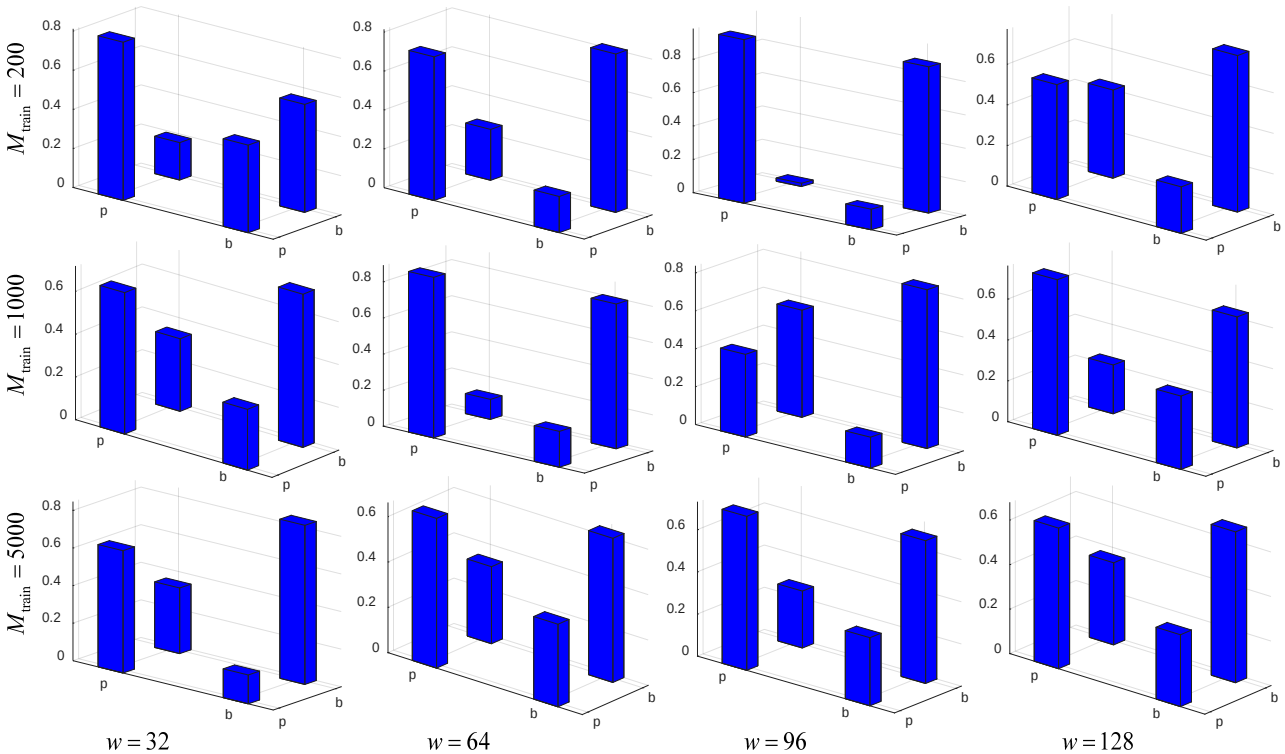


Fig. 6. An evolution of the normalized confusion matrix

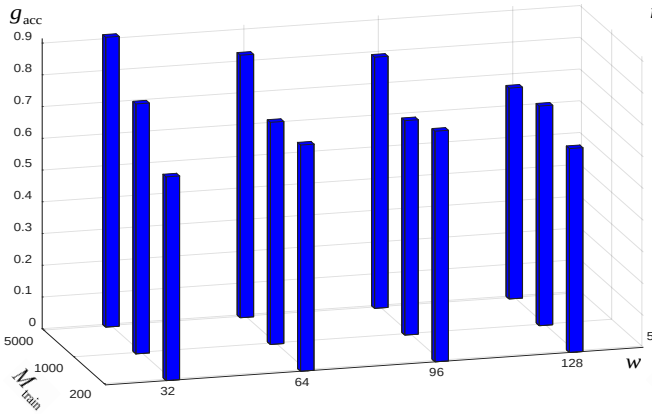


Fig. 7. The global accuracy

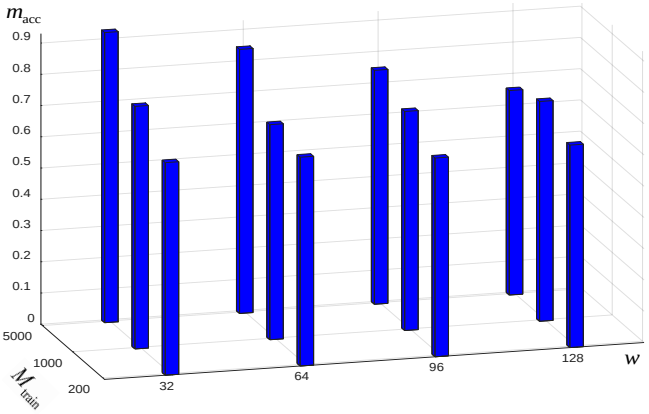


Fig. 8. The mean accuracy

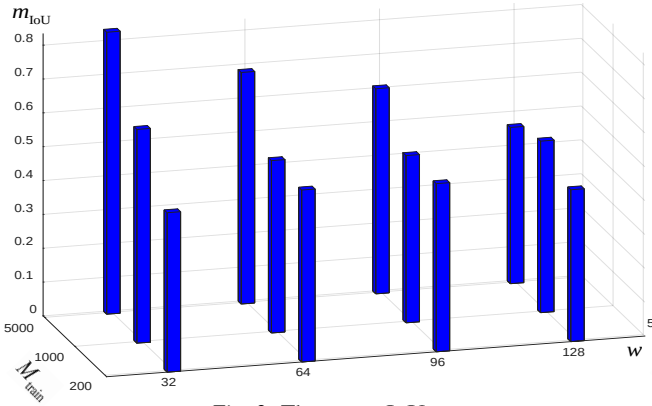


Fig. 9. The mean IoU

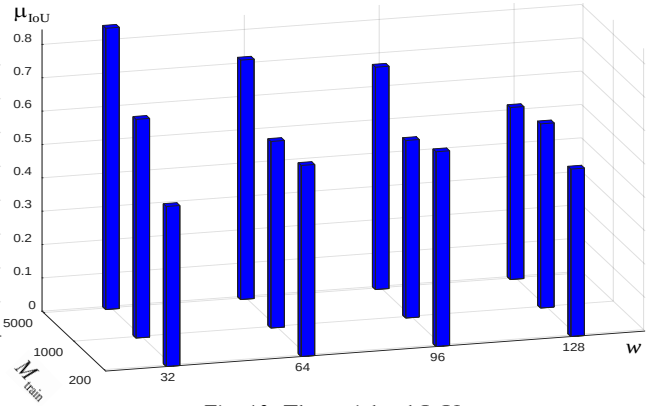
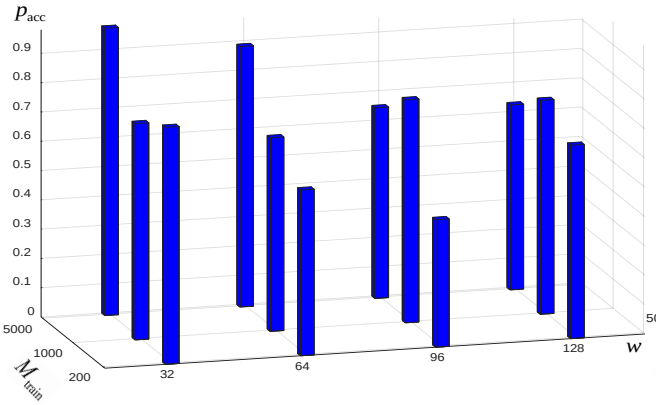
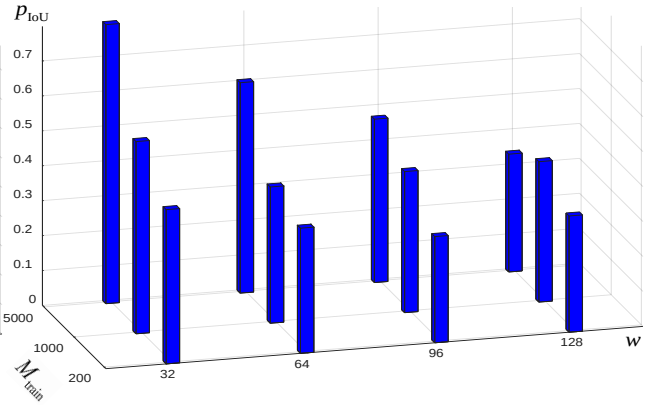


Fig. 10. The weighted IoU





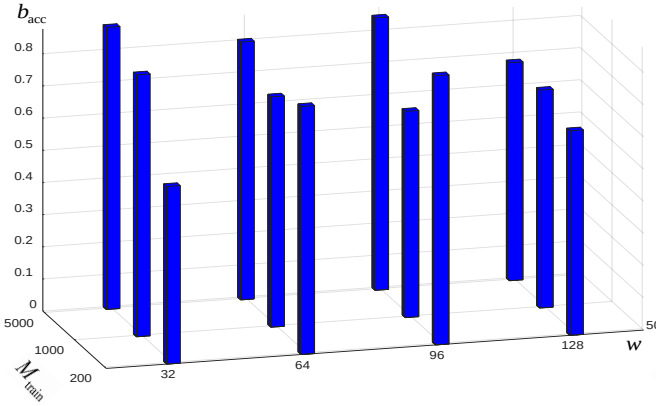
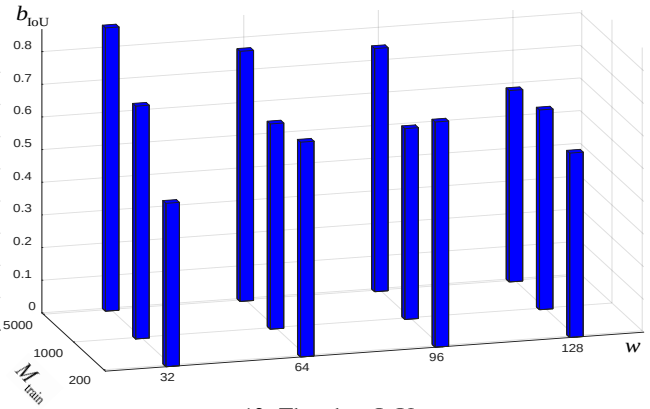


Fig. 11. The class accuracies



12. The class IoUs

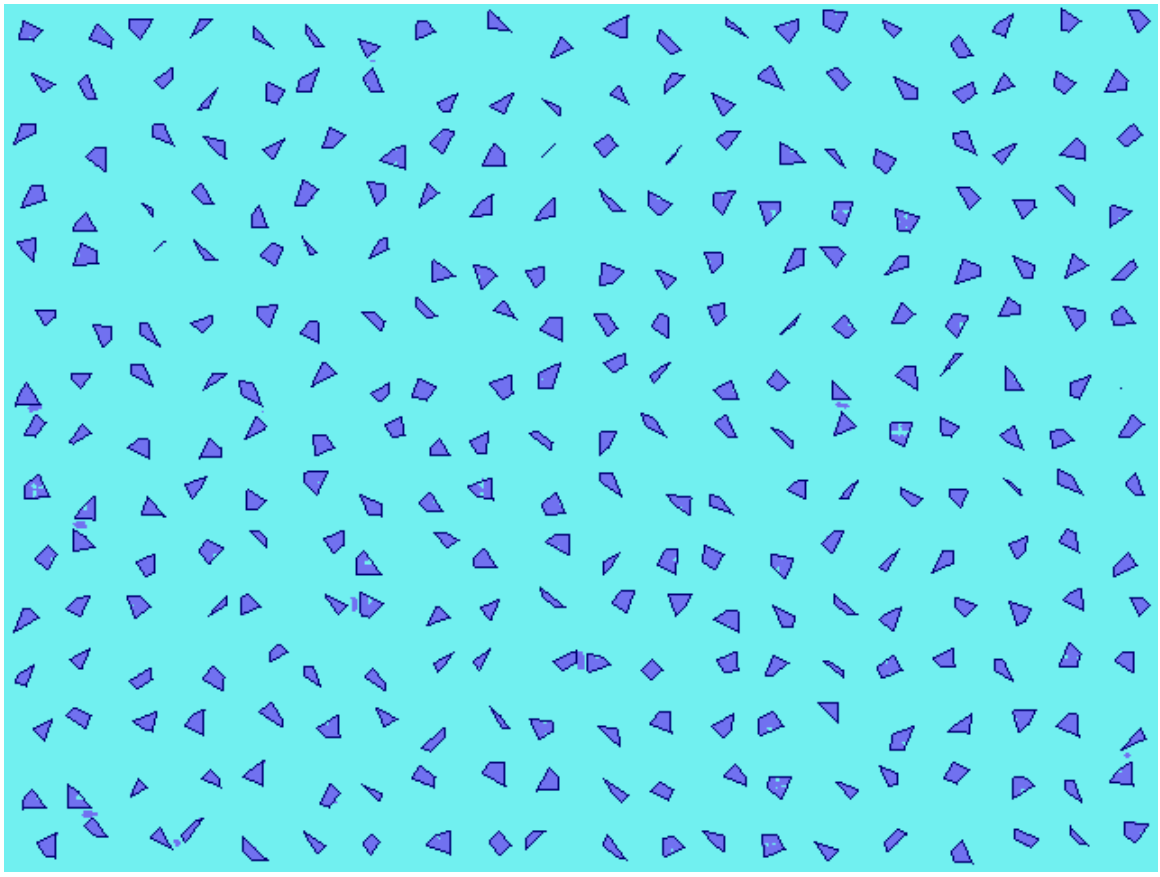


Fig. 13. The stack of 300 single-polygon images and the fused overlay image as a result of segmentation

ing another triple of a convolutional layer, ReLU and max pooling layer, followed by the second deconvolutional layer, the segmentation quality becomes improved.

The semantic segmentation network can process images that are larger than the specified input size. The smallest image size the network can process is $w \times w$. Besides, an input image may contain not only a single object. So, testing the trained network on a stack of polygons is possible. Fig. 13 shows how well the trained network performs on such a stack of 32×32 single-polygon images. The factual segmentation quality is pretty good but it may be a little worse than that dealing with a single image.

Such stacks are a more real model of situations rather than just a single object to be segmented. Here, the network full connectivity gives an opportunity to work with a great deal of image sizes starting off the minimal size 32×32. The input image thus is not necessary to be a square. The input square images used for training and testing the network are atomic instances to project it. If an input image, whichever polygonal objects it has, does not have size $(32q) \times (32t)$ by $q \in \mathbb{N}$, $t \in \mathbb{N}$, then it is just scaled (resized or adjusted) to the nearest size $(32q) \times (32t)$.

Discussion

Although any dataset generated by (1)–(6) is far away from a real-world task, it is a fast test platform that allows to adjust a segmentation network to the image size and its complexity. The simplest dataset is of triangles. When a polygon has four edges or more, it is harder to segment. Meanwhile, the larger the polygon, the harder it is segmented. The central part of the polygon is segmented poorer when the image size grows along with enlarging the polygon. This happens due to the polygon edges become relatively thinner, and thus the network "sees" them less legibly. Fuzzy border transitions additionally hamper the segmentation.

Larger images have polygons with the same thinnest edges, so it is not a matter of just scaling the smallest images. The larger the image size, the clearer bottlenecks of a network architecture can be seen. These bottlenecks are gradually eliminated

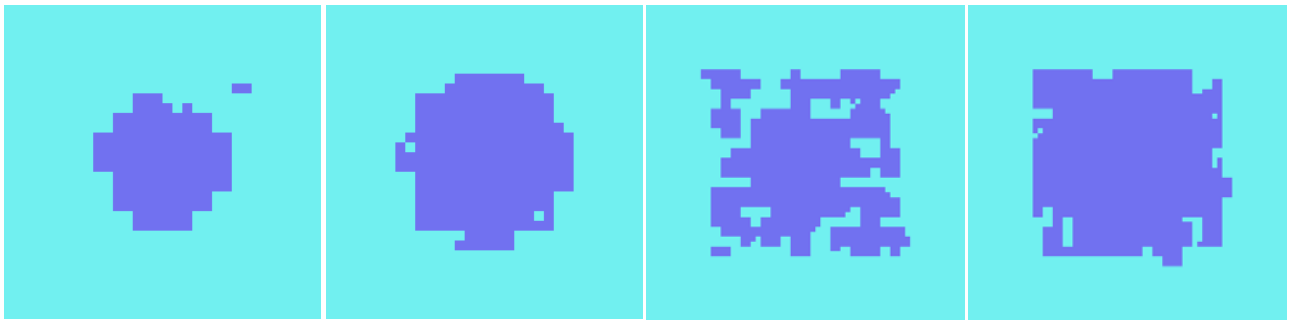


Fig. 14. Four examples of when the network segmented an empty image, that is an evidence of the failed training (similar to overfitting but not exactly it); note how the segmented area becomes more square-like as the training result worsens

by inserting more convolutional, deconvolutional, and max pooling layers. Numbers of filters are increased along with that. ReLUs and DropOut layers are inserted appropriately [6, 10].

Note that a segmentation network trained by a dataset generated by (1)–(6) must not "see" background itself. This is so because the dataset does not contain images with only class "background". Therefore, if the network being currently trained starts segmenting empty images (images of appropriate sizes regarding the network input, which do not contain any objects), the training process should be stopped (Fig. 14). The saved last version of the network not segmenting empty images becomes the best one for the given architecture and dataset. Further improvement of accuracy (or, in general, semantic segmentation quality by the said metrics) requires either modifications of the network architecture or generation of a dataset wherein polygons would be relatively smaller than previously.

Conclusions

The represented method of generating an infinitely scalable dataset relies on pseudorandomization of the polygon vertices' number by (1) and coordinates of the vertices by (2)–(4). Inequalities (5) and (6) help in making a polygon of an appropriate form and size, unless the polygon is a triangle. On rare occasions, the triangle can be generated very small or thin reminding an arrow (see Fig. 13).

The dataset generator is useful for rapidly obtaining a toy dataset of any volume and image size from scratch, careless of how to make photos, process them, sort and label them. The latter is the most important. Automatic labeling saves much time and resources. The dataset does not need augmentation. Despite primitivism of its objects, their transparency and thin border produce a "masking" effect, especially for larger-sized images. An impact of the segmentation task is strengthened by irregularity of the object's form. Finally, scalability and randomized positioning of the object's center of mass make the toy dataset an ideal means for testing segmentation network architectures much faster. Undesirable segmentation of empty images exemplified in Fig. 14 is an additional criterion of early stopping. Moreover, this may be an evidence of the poorly generated dataset (e. g., too large polygons like those in Fig. 2).

A further research based on such an infinite artificial dataset can be pursued with images containing two and more polygons. Sizes of the polygons and their ratios will vary randomly. An option of whether polygons can overlap or not will be included. If overlapping is admitted, polygonal edges of the overlap region disappear. In such a case, non-convex polygonal objects will be considered.

References

[1] H.-J. He *et al.*, "Image segmentation techniques", in *Computer Vision Technology for Food Quality Evaluation*, 2nd ed., D.-W. Sun, Ed. San Diego: Academic Press, 2016, pp. 45–63. doi: 10.1016/B978-0-12-802232-0.00002-5

[2] J. Rogowska, "Overview and fundamentals of medical image segmentation", in *Handbook of Medical Image Processing and Analysis*, 2nd ed., I.N. Bankman, Ed. San Diego: Academic Press, 2009, pp. 73–90. doi: 10.1016/B978-012373904-9.50013-1

[3] J.-T. Chien, "Deep neural network", in *Source Separation and Machine Learning*, J.-T. Chien, Ed. Academic Press, 2019, pp. 259–320. doi: 10.1016/B978-0-12-804566-4.00019-X

[4] V. Badrinarayanan *et al.*, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, iss. 12, pp. 2481–2495, 2017. doi: 10.1109/TPAMI.2016.2644615

[5] V.V. Romanuke, “Smooth non-increasing square spatial extents of filters in convolutional layers of CNNs for image classification problems”, *Appl. Comp. Syst.*, vol. 23, pp. 52–62, 2018. doi: 10.2478/acss-2018-0007

[6] V.V. Romanuke, “Appropriate number and allocation of ReLUs in convolutional neural networks”, *Naukovi Visti NTUU KPI*, no. 1, pp. 69–78, 2017. doi: 10.20535/1810-0546.2017.1.88156

[7] F.C. Pereira and S.S. Borysov, “Machine learning fundamentals”, in *Mobility Patterns, Big Data and Transport Analytics*, C. Antoniou *et al.*, eds. Elsevier, 2019, pp. 9–29. doi: 10.1016/B978-0-12-812970-8.00002-6

[8] V.V. Romanuke, “An attempt of finding an appropriate number of convolutional layers in CNNs based on benchmarks of heterogeneous datasets”, *Electrical, Control and Communication Engineering*, vol. 14, pp. 51–57, 2018. doi: 10.2478/ecce-2018-0006

[9] V.V. Romanuke, “Appropriate number of standard 2×2 max pooling layers and their allocation in convolutional neural networks for diverse and heterogeneous datasets”, *Inform. Technol. Manag. Sci.*, vol. 20, pp. 12–19, 2017. doi: 10.1515/itms-2017-0002

[10] V.V. Romanuke, “Appropriateness of DropOut layers and allocation of their 0.5 rates across convolutional neural networks for CIFAR-10, EEACL26, and NORB datasets”, *Appl. Comp. Syst.*, vol. 22, pp. 54–63, 2017. doi: 10.1515/acss-2017-0018

[11] J.-J. Lv *et al.*, “Data augmentation for face recognition”, *Neurocomputing*, vol. 230, pp. 184–196, 2017. doi: 10.1016/j.neucom.2016.12.025

[12] D. Avis *et al.*, “Simple on-line algorithms for convex polygons”, in *Machine Intelligence and Pattern Recognition*, vol. 2, G.T. Toussaint, Ed. North-Holland, 1985, pp. 23–42. doi: 10.1016/B978-0-444-87806-9.50007-4

[13] E. Horowitz and M. Papa, “Polygon clipping: Analysis and experiences”, in *Theoretical Studies in Computer Science*, J.D. Ullman, Ed. Academic Press, 1992, pp. 315–339. doi: 10.1016/B978-0-12-708240-0.50016-2

[14] R.L. Bowman, “Evaluating pseudo-random number generators”, in *Chaos and Fractals*, C.A. Pickover, Ed. Elsevier Science, 1998, pp. 133–142. doi: 10.1016/B978-044450002-1/50023-0

[15] T. Bradley *et al.*, “Parallelization techniques for random number generators”, in *GPU Computing Gems Emerald Edition. Applications of GPU Computing Series*, W.W. Hwu, Ed. Morgan Kaufmann, 2011, pp. 231–246. doi: 10.1016/B978-0-12-384988-5.00016-4

[16] J. Jackman, “The basics of how compositing works”, in *Bluescreen Compositing*, J. Jackman, Ed. Oxford, UK: Focal Press, 2007, pp. 231–246. doi: 10.1016/B978-1-57820-283-6.50004-8

[17] T. Praczyk, “A quick algorithm for horizon line detection in marine images”, *J. Marine Sci. Technol.*, vol. 23, iss. 1, pp. 164–177, 2018. doi: 10.1007/s00773-017-0464-8

В.В. Романюк

НАБІР ДАНИХ НЕСКІНЧЕННОГО МАСШТАБУВАННЯ ІЗ ЗОБРАЖЕНЬ У ГРАДАЦІЯХ СІРОГО З ОДНИМ БАГАТОКУТНИКОМ ЯК ШВИДКА ТЕСТОВА ПЛАТФОРМА ДЛЯ СЕМАНТИЧНОЇ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

Проблематика. Кожна нова задача семантичної сегментації зображень вимагає тонкої настройки архітектури мережі сегментації, яку дуже важко виконати на зображеннях високої роздільної здатності, що можуть містити багато категорій і залучати величезні обчислювальні ресурси. Таким чином, питання полягає в тому, чи можна набагато швидше протестувати архітектури мереж сегментації, щоб знайти оптимальні рішення, які могли б бути застосовані до реальних завдань семантичної сегментації зображень.

Мета дослідження. Стаття присвячена розробці набору даних із нескінченним масштабуванням, який міг би слугувати тестовою платформою для семантичної сегментації зображень. Набір даних буде містити будь-яку кількість записів будь-якого розміру, необхідного для тестування.

Методика реалізації. Новий штучний набір даних проектується для семантичної сегментації зображень. Цей набір даних являє собою зображення в градаціях сірого з білим тлом. Багатокутний об’єкт випадковим чином розміщується на тлі. Ребра багатокутника – чорні, а тіло багатокутника – прозоре. Таким чином, зображення набору даних являє собою набір ребер опуклого багатокутника на білому тлі. Край багатокутника має товщину в один піксель, але перехід між білим тлом і чорними краями багатокутника включає сірі пікселі в околиці однопіксельних країв. Такий шум є наслідком процесу перетворення формату файлу зображення. Кількість ребер багатокутника генерується випадковим чином для кожного наступного зображення. Розмір багатокутника і положення його центра мас щодо полів зображення також рандомізовані.

Результати дослідження. Модельний набір даних будь-якого обсягу і розміру зображення може бути згенерований “з нуля”. Крім того, генератор наборів даних автоматично категоризує пікселі для класів “background” і “polygon”. Такий набір даних не потребує збільшення. Зрештою, цей набір даних можна нескінченно масштабувати, і він буде слугувати платформою швидкого тестування для архітектур мереж сегментації.

Висновки. Розглянуті приклади використання набору даних із багатокутників підтверджують його придатність і здатність навчених мереж успішно сегментувати комплекти об’єктів. Крім того, виявляється критерій ранньої зупинки на основі сегментації порожнього зображення.

Ключові слова: семантична сегментація зображень; набір даних; багатокутний об’єкт; прозоре тло; збільшення; архітектура мережі сегментації; сегментація порожнього зображення.

В.В. Романюк

БЕСКОНЕЧНО МАСШТАБИРУЕМЫЙ НАБОР ДАННЫХ ИЗ ИЗОБРАЖЕНИЙ В ГРАДАЦИЯХ СЕРОГО С ОДНИМ МНОГОУГОЛЬНИКОМ КАК БЫСТРАЯ ТЕСТОВАЯ ПЛАТФОРМА ДЛЯ СЕМАНТИЧЕСКОЙ СЕГМЕНТАЦИИ ИЗОБРАЖЕНИЙ

Проблематика. Каждая новая задача семантической сегментации изображений требует тонкой настройки архитектуры сети сегментации, которую очень трудно выполнить на изображениях высокого разрешения, которые могут содержать много категорий и вовлекать огромные вычислительные ресурсы. Таким образом, вопрос заключается в том, можно ли гораздо быстрее протестировать архитектуры сетей сегментации, чтобы найти оптимальные решения, которые могли бы быть применены к реальным задачам семантической сегментации изображений.

Цель исследования. Статья посвящена разработке бесконечно масштабируемого набора данных, который мог бы служить тестовой платформой для семантической сегментации изображений. Набор данных будет содержать любое количество записей любого размера, необходимого для тестирования.

Методика реализации. Новый искусственный набор данных проектируется для семантической сегментации изображений. Этот набор данных представляет собой изображения в градациях серого с белым фоном. Многоугольный объект случайным образом размещается на фоне. Ребра многоугольника – черные, а тело многоугольника – прозрачное. Таким образом, изображение набора данных представляет собой набор ребер выпуклого многоугольника на белом фоне. Край многоугольника имеет толщину в один пиксель, но переход между белым фоном и черными краями многоугольника включает серые пиксели в окрестности однопиксельных краев. Такой шум является последствием процесса преобразования формата файла изображения. Количество ребер многоугольника генерируется случайным образом для каждого следующего изображения. Размер многоугольника и положение его центра масс относительно полей изображения также рандомизированы.

Результаты исследования. Модельный набор данных любого объема и размера изображения может быть сгенерирован “с нуля”. Кроме того, генератор наборов данных автоматически категоризирует пиксели для классов “background” и “polygon”. Такой набор данных не нуждается в приращении. В конце концов, этот набор данных можно бесконечно масштабировать, и он будет служить платформой быстрого тестирования для архитектур сетей сегментации.

Выводы. Рассмотренные примеры использования набора данных из многоугольников подтверждают его пригодность и способность обученных сетей успешно сегментировать комплекты объектов. Кроме того, обнаруживается критерий ранней остановки на основе сегментации пустого изображения.

Ключевые слова: семантическая сегментация изображений; набор данных; многоугольный объект; прозрачный фон; приращение; архитектура сети сегментации; сегментация пустого изображения.

Рекомендована Радою
факультету прикладної математики
КПІ ім. Ігоря Сікорського

Надійшла до редакції
13 грудня 2018 року

Прийнята до публікації
28 лютого 2019 року