Applied Computer Systems
ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
May 2020, vol. 25, no. 1, pp. 62–69
https://doi.org/10.2478/acss-2020-0008
https://content.sciendo.com

A Prototype Model for Semantic Segmentation of Curvilinear Meandering Regions by Deconvolutional Neural Networks

Vadim Romanuke*
Polish Naval Academy, Gdynia, Poland

***Abstract* – Deconvolutional neural networks are a very accurate tool for semantic image segmentation. Segmenting curvilinear meandering regions is a typical task in computer vision applied to navigational, civil engineering, and defence problems. In the study, such regions of interest are modelled as meandering transparent stripes whose width is not constant. The stripe on the white background is formed by the upper and lower non-parallel black curves so that the upper and lower image parts are completely separated. An algorithm of generating datasets of such regions is developed. It is revealed that deeper networks segment the regions more accurately. However, the segmentation is harder when the regions become bigger. This is why an alternative method of the region segmentation consisting in segmenting the upper and lower image parts by subsequently unifying the results is not effective. If the region of interest becomes bigger, it must be squeezed in order to avoid segmenting the empty image. Once the squeezed region is segmented, the image is conversely rescaled to the original view. To control the accuracy, the mean BF score having the least value among the other accuracy indicators should be maximised first.**

***Keywords* – Curvilinear meandering region, deconvolutional layer, empty image segmentation, mean BF score, neural network, overfitting, semantic segmentation, toy dataset.**

I. Introduction

In computer vision, semantic image segmentation (SIS) is a problem of labelling specific regions of an image or a series of video frames [1], [2]. This problem has been intensively studied since the 2010s. However, methods and tools for SIS have been developed by experiencing practical tasks rather than based on theoretic aspects and principles [1], [3], [4]. Thus, as of 2020, still no unified theory of SIS is built.

Typical tasks solved by SIS arise in autopilot-induced driving, traffic engineering, surveillance systems, marine prospecting and defence, medicine, geological exploration, geothermal prospecting and other associated fields where the tasks are to spot and control definite objects or regions, or to retrieve useful information from the image or video [2], [4], [5]. The potential application of SIS will probably have an impact on civil engineering, metallurgy, highway engineering, rescue operations, and microbiology.

The simplest method of SIS is the thresholding mostly suitable for grayscale images in order to label plain regions [1]. Colour images can also be thresholded, where a separate threshold for each of the RGB components of the image is designated, and then these three are combined with logical conjunction [1]. More complicated SIS is fulfilled with clustering methods [6], histogram-based methods [7], and methods based on solving partial differential equations [8]. The most promising approach to SIS is in developing neural networks whose performance is close to perfect if they are properly trained. However, one of the main open problems of SIS is the absence of a general routine to solve a specific task. Once the task is described, a heuristic process of choosing an approach and its parameters starts [1], [2], [7].

Commonly, a neural network for SIS named the semantic segmentation network (SSN) classifies every pixel in an image [5], [9], so the network processes huge amounts of data. The SSN architecture is based on an encoder/decoder structure [2], [9] consisting of three parts: a downsampling subnetwork, an upsampling subnetwork, and a pixel classification layer. A downsampling subnetwork is stacked of convolutional layers (ConvLs), ReLUs, and max pooling layers. An upsampling subnetwork is stacked of deconvolutional layers (DeConvLs) and ReLUs. The final layer starts with a set of 1-by-1 ConvLs (which is a fully-connected layer), whose number is equal to the number of classes, followed by the softmax and pixel classification layers which categorically label each image pixel [10]. Notwithstanding the relative simplicity of the SSN architecture, specification of its parameters and hyperparameters is not a trivial task.

II. Motivation

Effective SIS requires optimally configured SSNs. Optimal SSN configuration should not be based on rules of thumb, which do not always fit real-world problems. It should be rather based on a theoretically sound prototyping using toy datasets [2]. The goal of the prototyping is to quickly build an SSN capable of segmenting successfully real-world images by using artificially created images containing generalized objects or regions of interest [3], [5], [8]. At this stage, SSNs are trained and tested on a toy dataset whose volume is unlimited and image size is small that allows quickly trying different SSN configurations.

* Corresponding author's e-mail: romanukevadimv@gmail.com

One of the attempts to supplement the bank of toy datasets was in article [10], which presented a toy dataset generator for prototyping SSNs capable of segmenting real-world complex objects. The objects were modelled by convex polygons. The complexity of polygons was suggested to be regulated by the number of edges in a polygon, the maximal number of polygons in one image, the set of scale factors, and the set of probabilities determining how many polygons in a current image are generated. As a result of the attempt, a tool for prototyping SSNs capable of segmenting real-world objects without strong curvature (vehicles, buildings, playgrounds, people at distance, etc.) was developed.

Nevertheless, there are many practical examples where objects (or regions) have strong curvature. For instance, they are landscape scenes whose segmentation is a basis of applying computer vision to navigational, civil engineering, and defence problems [1], [2], [7], [8]. One of the primary tasks is to see the land line, riverbed, highway, etc. (Fig. 1). Such regions can hardly be modelled by chains of tiny triangles and polygons (as an approximation of their curvature), so a toy dataset generator of curvilinear meandering regions is needed.

Fig. 1. Examples of regions with strong curvature to be segmented. Often such objects can divide the image in two parts (upper and lower) like the coast separates the sky from the seawater (in the last two images in the third row).

Another issue is segmentation of the empty image, in which class "background" exists only [10], [11]. The empty image segmentation is similar to overfitting, but it is not exactly that. In the worst case, the curvilinear meandering region separates the upper image part from the lower image part, so this may be a challenge as the segmentation is harder for bigger image parts due to a risk of segmenting class "background" [11].

It is worth noticing that the case with the complete separation of the upper and lower image parts is not that easy as it may seem. The matter is that the width of the curvilinear meandering region varies. Besides, the colour of the region interior can coincide with the outside colours. Thus, SIS can be fulfilled by thresholding only in peculiar cases.

III. Goal and Steps to Achieve It

Issuing from the lack of theoretic approaches to SIS of curvilinear meandering regions, the goal is to determine a better way to segment them by the SSN. To achieve the goal, the image size, a pattern of the regions and an algorithm of generating datasets are to be substantiated. Then, a few SNN configurations are to be tried in order to directly segment the regions. On the other hand, a method of indirect segmentation should be tried as well, in which the upper and lower image parts are directly segmented, and then the region is segmented by unifying the results. In the study, the case with the complete separation of the upper and lower image parts will be considered.

IV. Image Size and Regions to be Segmented

Training neural networks on toy dataset images, there are classical datasets like CIFAR-10, CIFAR-100, MNIST, NORB, EEACL26, etc. [11]. Researchers tend to set the image size at dimensions, which are 2 raised to some integer power [9] ensuring faster training owing to consistency with the binary system hardware, on which computational algorithms are physically implemented. Thus, the best image size is either 32×32 or 64×64. As SIS in real-world tasks deals with images of higher resolution, it is appropriate to use the 64×64 size for the further study.

The region of interest is a curvilinear meandering stripe whose width is not constant. The background is white. To simulate severer conditions of SIS, the stripe is formed by the upper and lower curves of the black colour. The lower curve (border) is obviously not parallel to the upper one. Moreover, the thickness of the curves will be just one pixel, and the interior of the stripe is white. In fact, the region of interest is 100 % transparent. This additionally must hamper the segmentation in order to ensure good generalization.

V. Algorithm of Generating Datasets

The dataset is of N images and N respective images with labelled regions [10], [11]. The image and its labelled version are formed by using the three input arguments: positive integers k_1 and k_2 defining the number of key points (denoted by k) along the horizontal axis, and a positive integer w_{str} defining the average width of the stripe. Integer

$$k = \psi\left(k_1 \cdot \Xi(1)\right) + k_2 \tag{1}$$

is a number of extrema along the upper and lower borders of the stripe, where $\Xi(K)$ is a vector (or a set) of K pseudorandom numbers drawn from the standard uniform distribution on interval $(0;1)$ and function $\psi(x)$ returns the integer part of number x (e. g., see [10]). Therefore, integer (1) takes values between k_2 and $k_1 - 1 + k_2$.

All key points along the horizontal axis constitute a set

$$X_{\text{sorted}} = \left\{1, \psi\left(63 \cdot \Xi(k) + 2\right), 64\right\} \tag{2}$$

whose $k+2$ elements are sorted in ascending order. Then the respective points along the vertical axis constitute a set

$$Y_{\text{up}} = -w_{\text{str}} + \psi\left(63 \cdot \Xi(k+2) + 1\right) \tag{3}$$

of $k+2$ elements without sorting. Then the preliminary 64

upper border points are found by the cubic spline data interpolation using the data values in (3) at the data sites in (2):

$$\tilde{Y}_{\text{up.border}} = \text{spline}_3\left(X_{\text{sorted}}, Y_{\text{up}}, \left\{\overline{1,\,64}\right\}\right). \quad (4)$$

The preliminary 64 lower border points are found similarly:

$$Y_{\text{low}} = Y_{\text{up}} + \psi\left(\left(4w_{\text{str}} - 1\right)\cdot \Xi\left(k+2\right) + 1\right) \quad (5)$$

and then

$$\tilde{Y}_{\text{low.border}} = \text{spline}_3\left(X_{\text{sorted}}, Y_{\text{lower}}, \left\{\overline{1,\,64}\right\}\right). \quad (6)$$

Elements in sets (4) and (6) are corrected and re-scaled to fit the 64×64 image frame. For this purpose, the range

$$r = \frac{\max \tilde{Y}_{\text{low.border}} - \min \tilde{Y}_{\text{up.border}}}{2} \quad (7)$$

and the average value

$$\bar{y} = \text{mean}\left(\frac{\tilde{Y}_{\text{low.border}} + \tilde{Y}_{\text{up.border}}}{2}\right) \quad (8)$$

are used. First,

$$\tilde{Y}^{(\text{obs})}_{\text{low.border}} = \tilde{Y}_{\text{low.border}}, \quad \tilde{Y}_{\text{low.border}} = \tilde{Y}^{(\text{obs})}_{\text{low.border}} - \bar{y}, \quad (9)$$

$$\tilde{Y}^{(\text{obs})}_{\text{up.border}} = \tilde{Y}_{\text{up.border}}, \quad \tilde{Y}_{\text{up.border}} = \tilde{Y}^{(\text{obs})}_{\text{up.border}} - \bar{y}, \quad (10)$$

whereupon a range corrector

$$r_{\text{corr}} = r \big/ \left[w_{\text{str}}\left(1 + 0.25\cdot\theta\right)\right] \quad (11)$$

is found, where θ is a pseudorandom number drawn from the standard normal distribution (with zero mean and unit variance). Then the second correction follows:

$$\tilde{Y}^{(\text{obs})}_{\text{low.border}} = \tilde{Y}_{\text{low.border}}, \quad \tilde{Y}_{\text{low.border}} = \tilde{Y}^{(\text{obs})}_{\text{low.border}} \big/ r_{\text{corr}}, \quad (12)$$

$$\tilde{Y}^{(\text{obs})}_{\text{up.border}} = \tilde{Y}_{\text{up.border}}, \quad \tilde{Y}_{\text{up.border}} = \tilde{Y}^{(\text{obs})}_{\text{up.border}} \big/ r_{\text{corr}}. \quad (13)$$

Coefficient

$$y_{\text{corr}} = 64 \big/ \left(2 + 0.25\cdot\theta\right) \quad (14)$$

is used for the third, final, correction:

$$\tilde{Y}^{(\text{obs})}_{\text{low.border}} = \tilde{Y}_{\text{low.border}}, \quad \tilde{Y}_{\text{low.border}} = \rho\left(\tilde{Y}^{(\text{obs})}_{\text{low.border}} + y_{\text{corr}}\right), \quad (15)$$

$$\tilde{Y}^{(\text{obs})}_{\text{up.border}} = \tilde{Y}_{\text{up.border}}, \quad \tilde{Y}_{\text{up.border}} = \rho\left(\tilde{Y}^{(\text{obs})}_{\text{up.border}} + y_{\text{corr}}\right), \quad (16)$$

where function $\rho(x)$ rounds x to the nearest integer.

Once the stripe borders are calculated by (15) and (16) as

$$\tilde{Y}_{\text{low.border}} = \left\{y^{<i>}_{\text{low}}\right\}_{i=1}^{64} \text{ and } \tilde{Y}_{\text{up.border}} = \left\{y^{<i>}_{\text{up}}\right\}_{i=1}^{64}, \quad (17)$$

they are set at the white 64×64 background, in which points

$$\left\{i,\, y^{<i>}_{\text{up}}\right\} \text{ and } \left\{i,\, y^{<i>}_{\text{low}}\right\} \text{ for every } i = \overline{1,\,64} \quad (18)$$

are set black. Thus, the image with a transparent stripe is formed with classes "stripe" and "background". The respective labelled image is the black 64×64 frame, at which the stripe is set: the vertical space between points (18) is filled with white.

By the given N, k_1, k_2, and w_{str}, datasets are generated by repeating routine (1)–(18) N times. An example of the generation for $k_1 = 2$, $k_2 = 1$, $w_{\text{str}} = 16$ (the stripe can have two bends at the most) is shown in Fig. 2. An example of a dataset with "more" curvilinear and narrower instances by $k_1 = 5$, $k_2 = 1$, $w_{\text{str}} = 8$ is presented in Fig. 3.

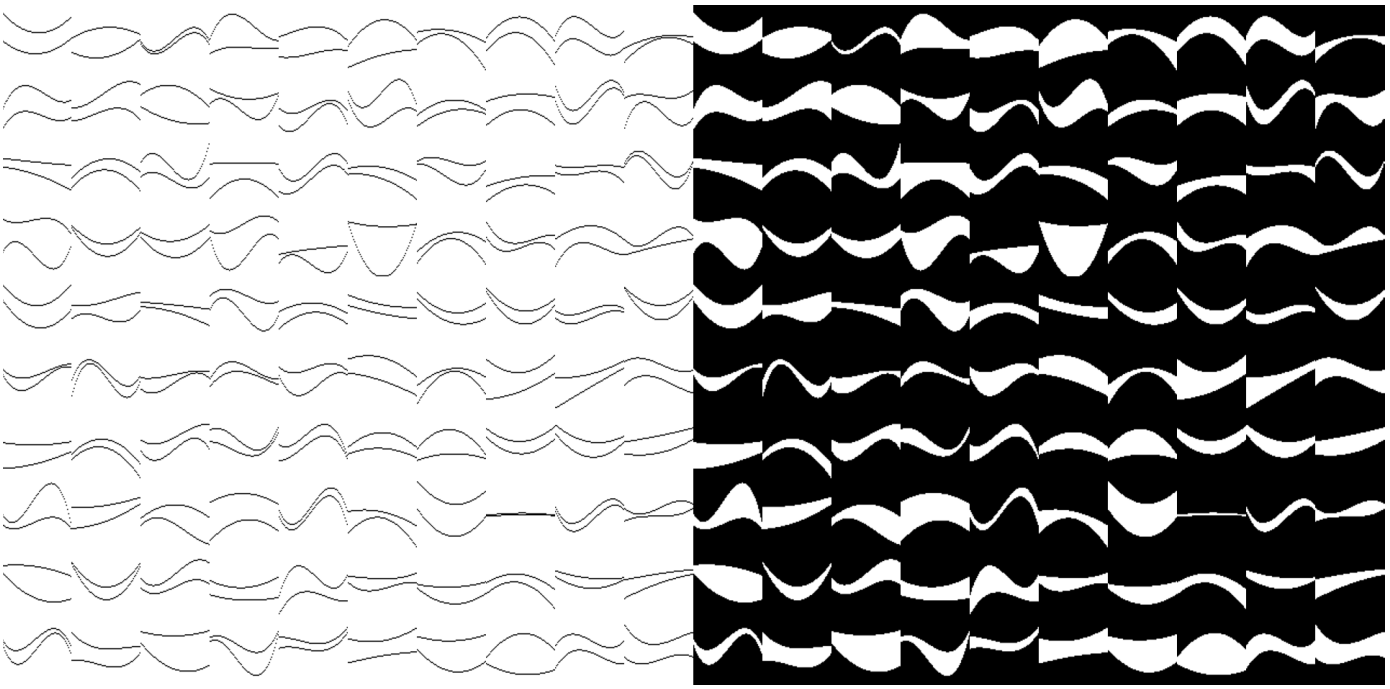

Fig. 2. Part of an artificial dataset of curvilinear meandering regions (grayscale image left and labels right), where the regions are very simple (the stripe can have two bends at the most). The stripes completely separate the upper and lower image parts. Such regions seem quite easy to be segmented (without SSNs). The most instances have pretty thick regions. However, some instances appear as toy dataset artefacts, where the region is either extremely thin (see the third image from the right in the third row from the bottom) or too thick (see the sixth image from the left in the fourth row from the top).

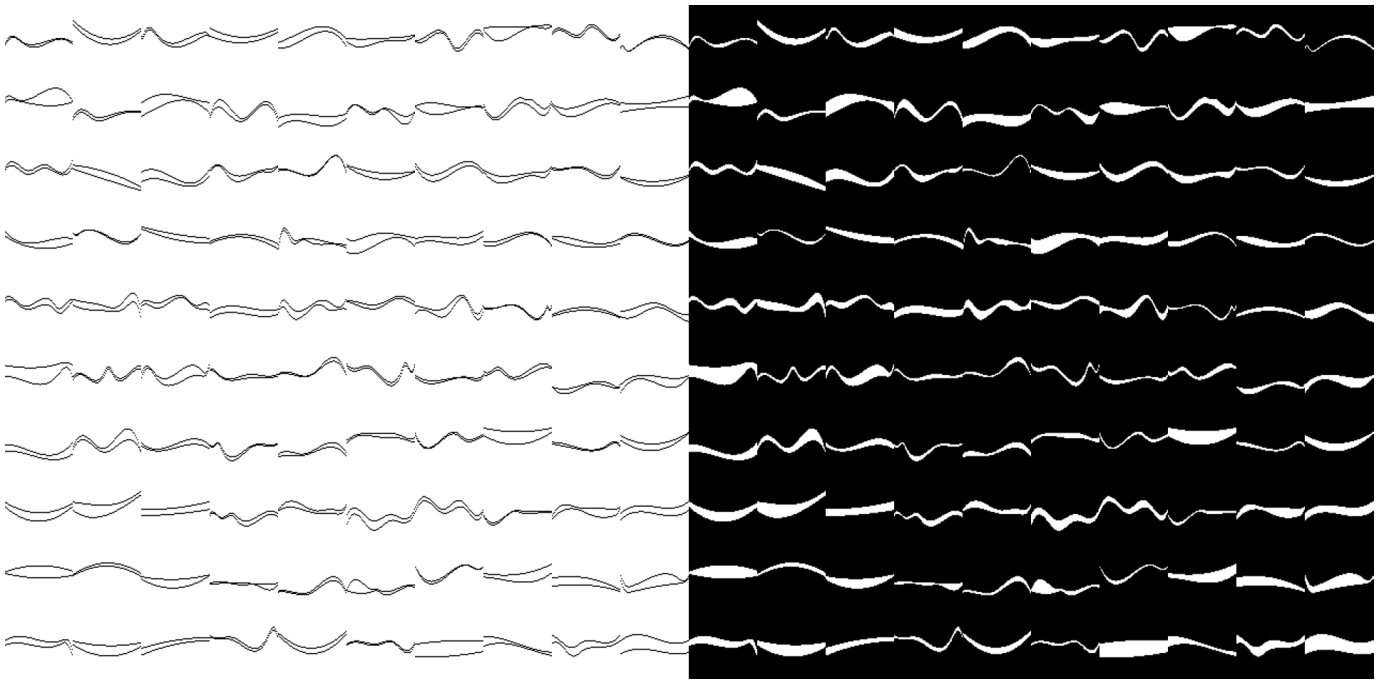

Fig. 3. Part of an artificial dataset of curvilinear meandering regions, where the regions appear "more" curvilinear (the stripe can have one to five bends) yet narrower than in Fig. 2. The seeming simplicity for SIS is ignored because the complete separation of the image by the region of interest cannot be known or predicted in advance, and thus the 100 % transparency of the region may become a serious difficulty in fulfilling SIS. Such datasets will be used for the study.

VI. Structure of Experiments

For the experimental study, datasets of 1000, 5000, and 25 000 instances are generated by $k_1 = 5$, $k_2 = 1$, $w_{\text{str}} = 8$ (see the example of the instances in Fig. 3). The best starting SSN configuration is that with two ConvLs and single DeConvL (the fully-connected layer before the softmax layer is not counted) used in [10], [11], where the number of filters can be set at 64, 128, 192, 256. This configuration can be deepened by adding ConvLs into the downsampling subnetwork and the respective DeConvLs into the upsampling subnetwork. The option of inserting a DropOut layer is also included. Eventually, methods of direct and indirect segmentation will be tried on those configurations.

VII. Training, Validation, and Testing

The SSN is trained on the dataset by the following training parameters: the initial learning rate is 0.001, weight decay is 0.0005, mini-batch size is 8, learning rate drop factor is 0.975, learning rate drop period $g_{\text{LRD}} = 10$, momentum is 0.9 (this is 90 % contribution of the parameter update of the previous iteration to the current one). Thus, 85 % of the dataset instances are used for training, and 15 % are used for validation. The SSN is trained by a routine that allows avoiding the segmentation of empty images [10], [11]:

1. Train for an epoch.
2. Set the current number of extra epochs to zero ($c_{\text{AE}} = 0$).
3. While the number of segmented pixels in the empty image for class "stripe" is zero and $c_{\text{AE}} < c_{\max}$ do:
   3.1. Drop the learning rate after every new g_{LRD} epochs.
   3.2. Train.
   3.3. Increase c_{AE} by 1.

Once an SSN is successfully trained, it is tested. For testing the trained SSN, a test dataset of 200 instances is formed. The performance of the SSN is estimated by its accuracy calculated along with the intersection-over-union (IoU) [11].

VIII. Results of Direct SIS

Raw experiments (by $c_{\max} < 10$) confirm that using a dataset of a greater volume gives slightly higher accuracy. Therefore, the dataset with 25000 instances is subsequently used (all the more, this dataset is artificial and it can be generated as voluminous as needed). Besides, the number of filters set at 128 (just twice as greater as the image size) appears to be better than 64 and not worse than 192 or 256. As the SSN performs slower with increasing the number of filters, the amount in 128 filters is approximately optimal here as for a shallow SSN architecture (Fig. 4), as well as for a deeper one (Fig. 5).

```
 1 Image Input              64x64x1 images
                            with 'zerocenter' normalization
 2 Convolution #1           128 3x3x1 convolutions
                            with stride [1  1] and padding [1  1  1  1]
 3 ReLU #1                  ReLU
 4 Max Pooling              2x2 max pooling with stride [2  2]
                            and padding [0  0  0  0]
 5 Convolution #2           128 3x3x128 convolutions
                            with stride [1  1] and padding [1  1  1  1]
 6 ReLU #2                  ReLU
 7 Transposed Convolution   128 4x4x128 transposed convolutions
                            with stride [2  2] and output cropping [1  1]
 8 Dropout                  50% dropout
 9 Convolution #3           2 1x1x128 convolutions
                            with stride [1  1] and padding [0  0  0  0]
10 Softmax                  softmax
11 Pixel Classification Layer   Class weighted cross-entropy loss
                                with classes 'spline_obj' and 'background'
```

Fig. 4. The SSN architecture (in MATLAB) with two ConvLs and a single DeConvL. The DropOut layer following the DeConvL is optional.

```
 1 Image Input                 64x64x1 images
                               with 'zerocenter' normalization
 2 Convolution #1              128 3x3x1 convolutions with stride [1  1]
                               and padding [1  1  1  1]
 3 ReLU #1                     ReLU
 4 Max Pooling #1              2x2 max pooling with stride [2  2]
                               and padding [0  0  0  0]
 5 Convolution #2              128 3x3x128 convolutions with stride [1  1]
                               and padding [1  1  1  1]
 6 ReLU #2                     ReLU
 7 Max Pooling #2              2x2 max pooling with stride [2  2]
                               and padding [0  0  0  0]
 8 Convolution #3              128 3x3x128 convolutions with stride [1  1]
                               and padding [1  1  1  1]
 9 ReLU #3                     ReLU
10 Transposed Convolution #1   128 4x4x128 transposed convolutions
                               with stride [2  2] and output cropping [1  1]
11 Transposed Convolution #2   128 4x4x128 transposed convolutions
                               with stride [2  2] and output cropping [1  1]
12 Dropout                     50% dropout
13 Convolution #4              2 1x1x128 convolutions with stride [1  1]
                               and padding [0  0  0  0]
14 Softmax                     softmax
15 Pixel Classification Layer   Class weighted cross-entropy loss
                                with classes 'spline_obj' and 'background'
```

Fig. 5. The SSN architecture (in MATLAB) with three ConvLs and two DeConvLs. The DropOut layer following the last DeConvL is optional.

The SSN is trained on the dataset whose regions have apparent curvilinearity and simultaneously are narrow with respect to the image size (see Fig. 3). The accuracy of the SSN performance is estimated by the five indicators: global accuracy (the ratio of correctly classified pixels, regardless of class, to the total number of pixels), mean accuracy (it regards classes), mean IoU (the ratio of correctly classified pixels to the total number of ground truth and predicted pixels), weighted IoU (it weights by the number of pixels in each class), and mean BF score (boundary/contour matching score between the predicted and true segmentation) [7], [11].

Obviously, SSNs by Fig. 4 are trained faster than those by Fig. 5. An easier and faster way to SIS is studied first. Thus, an SSN trained by Fig. 4 without DropOut has seemingly good indicators of the performance (see Table I). Nevertheless, visualization of the test dataset of 200 instances segmented by this SSN (see Fig. 6) reveals that the segmentation is "dirty". Indeed, almost every instance is erroneously segmented on outside of a region. Besides, some region interiors of a greater size are not perfectly segmented and such gaps are easily seen inside the regions. Contrary to the high global and mean accuracies, this "dirty" segmentation is expressed in the comparatively low mean IoU (the weighted IoU is not very high as well) and mean BF score.

TABLE I
PERFORMANCE OF THE SSN TRAINED FOR 33 EPOCHS BY FIG. 4 WITHOUT DROPOUT

Global accuracy	Mean accuracy	Mean IoU	Weighted IoU	Mean BF score
0.97471	0.98197	0.86402	0.95544	0.82016

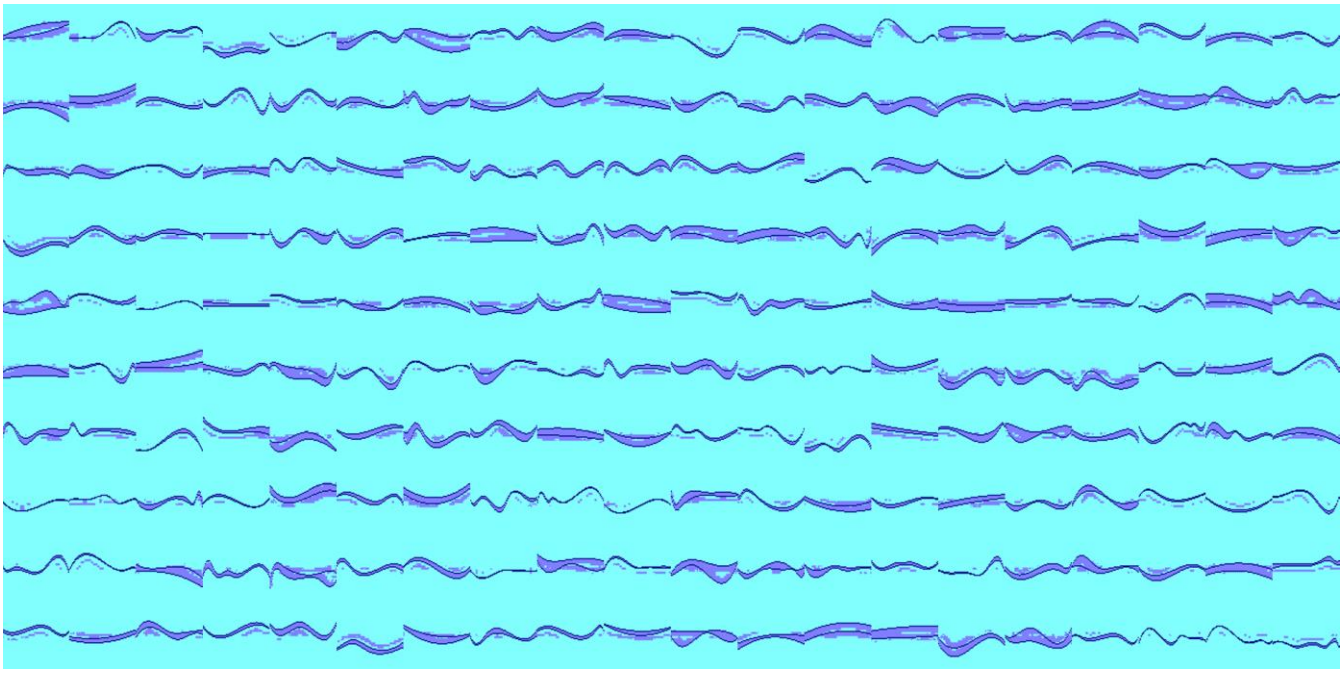

Fig. 6. The stack of 200 test images and the fused overlay image as a result of the direct SIS by the SSN trained for 33 epochs by Fig. 4 without a DropOut layer. The training is stopped due to the empty image segmentation. The global and mean accuracies here are not low at all but the mean IoU and mean BF score are really low (Table I). The weighted IoU is not low but the low mean IoU depreciates it anyway. The SSN "sees" thicker regions poorly leaving erroneously segmented pixels on outside of the regions and not segmenting their interiors entirely. Therefore, this segmentation result is too "dirty" and cannot be accepted.

When a DropOut layer is turned on, the SSN with the single DeConvL is trained a lit bit slower per epoch. The entire training process in epochs is slightly longer (now, it is 36 epochs). The SSN has better indicators of the performance (see Table II) but the mean BF score is still unsatisfactory. The segmentation result visualized in Fig. 7 still has "dirty" outcomes, although it is now clearer than that in Fig. 6.

TABLE II
PERFORMANCE OF THE SSN TRAINED FOR 36 EPOCHS BY FIG. 4

Global accuracy	Mean accuracy	Mean IoU	Weighted IoU	Mean BF score
0.98415	0.9828	0.90642	0.97076	0.85601

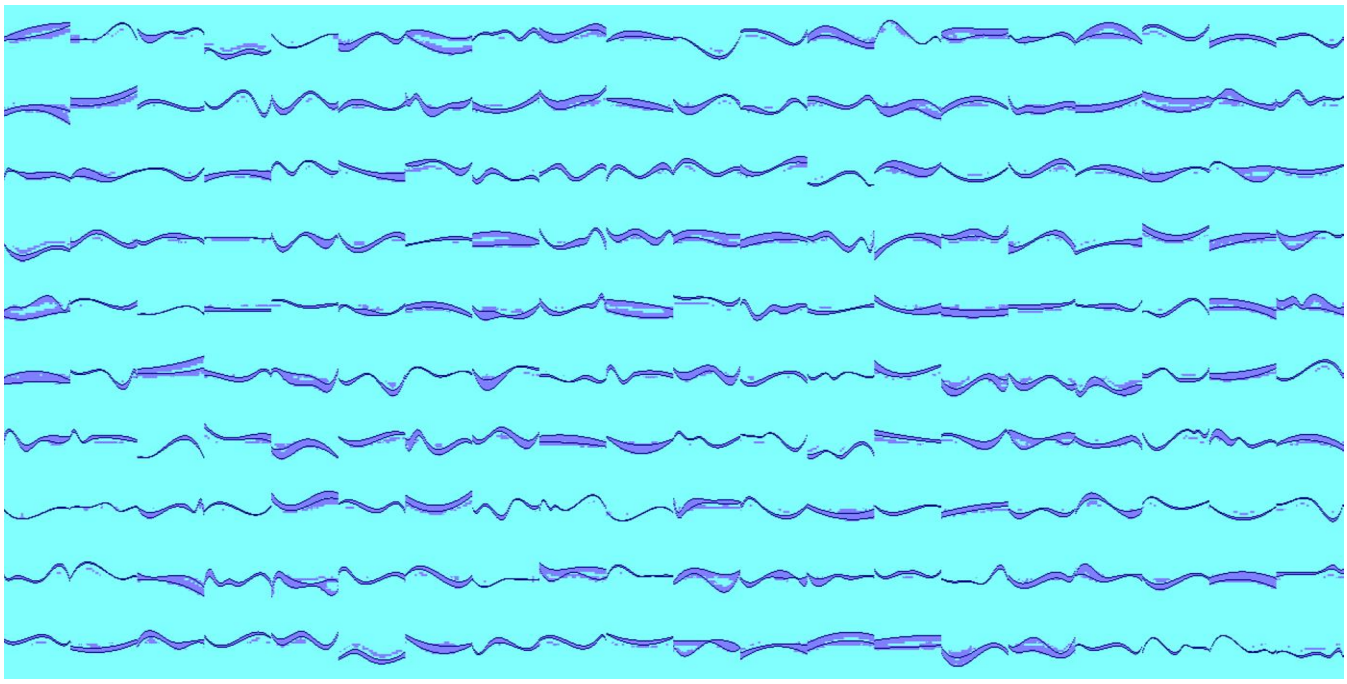

Fig. 7. The segmentation result after turning the DropOut layer on (following the single DeConvL) by Fig. 4. The SSN trained for 36 epochs starts segmenting the empty image and thus the training process is stopped. The global and mean accuracies here are a bit greater than those for the SSN without DropOut. Contrariwise, the mean IoU and mean BF score are significantly improved (Table II). The weighted IoU is improved as well. However, the mean BF score is still unsatisfactory. Although the SSN "sees" thicker regions better than the SSN without DropOut, the gaps inside their interiors did not disappear. The erroneously segmented pixels on outside of the regions are still clearly seen. Therefore, SIS by the shallow SSN architecture (Fig. 4) is still "dirty" and cannot be accepted.

As the segmentation results by the shallow SSN architecture (Fig. 4), whether DropOut is used or not, are far from perfect, the deeper SSN in Fig. 5 should be tried. Obviously, the SSN with an additional ConvL (and the respective second DeConvL) is trained slower per epoch. In this case, the entire training process lasts far longer (it is 150 epochs achieving the maximal number of extra epochs). However, the performance of the SSN trained by Fig. 5 without DropOut has almost perfect indicators (see Table III). Moreover, even training just for a third of this number of epochs gives indicators, which are very close to those in Table III. This is possible owing to the successful training at the very first epoch (see Fig. 8).

TABLE III

PERFORMANCE OF THE SSN TRAINED FOR 150 EPOCHS BY FIG. 5 WITHOUT DROPOUT

Global accuracy	Mean accuracy	Mean IoU	Weighted IoU	Mean BF score
0.99948	0.99961	0.99644	0.99896	0.99225

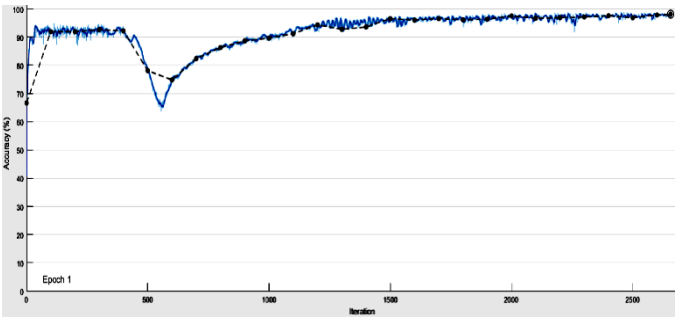

Fig. 8. The accuracy of the SSN (Fig. 5) without the DropOut layer trained during the first epoch, after which the validation accuracy achieves 98.11 %.

The test instance segmented worst by the deeper SSN without DropOut is shown in Fig. 9. Even here, it is very hard to notice any bad of SIS. It is worth noting that the region still has a varying width, although it is hardly noticeable. Obviously, such a segmentation result entirely visualized in Fig. 10 is quite acceptable.

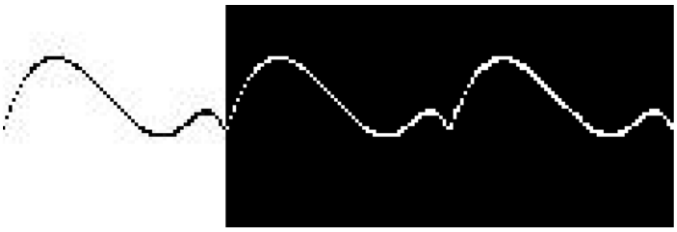

Fig. 9. The test instance (the test image versus truth and segmented images) segmented worst by the deeper SSN (Fig. 5) without DropOut trained for 150 epochs (whose performance indicators are presented in Table III). Visually, the erroneously segmented pixels (or non-segmented interior pixels) are hardly noticed. The region is too narrow, though (appearing roughly just as a one-pixel line rather than a region or stripe having some interior space).

As the deeper SSN without DropOut segments with a few flaws, it is reasonable to turn on the DropOut layer and see whether the performance could be perfected. Unexpectedly, the SSN by Fig. 5 performs slightly poorer (Table IV). Therefore, deeper SSNs without DropOut are the best for direct SIS by considering curvilinear meandering stripes.

TABLE IV

PERFORMANCE OF THE SSN TRAINED FOR 150 EPOCHS BY FIG. 5

Global accuracy	Mean accuracy	Mean IoU	Weighted IoU	Mean BF score
0.99868	0.99924	0.99108	0.99739	0.98373

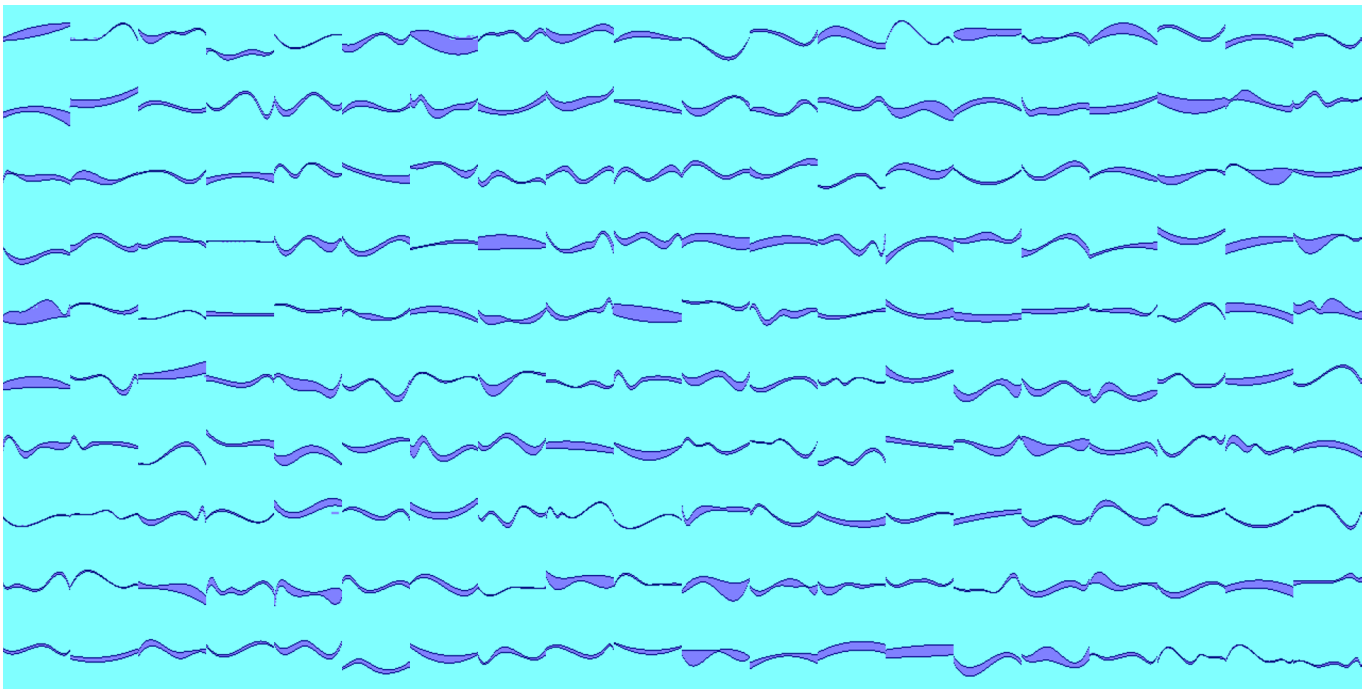

Fig. 10. The stack of 200 almost perfectly segmented test images and the fused overlay image as visualization of the performance of the SSN trained for 150 epochs by Fig. 5 without DropOut. Along with the test instance in Fig. 9, a few lacks can be seen (e. g., see the 7-th image with the thick stripe in the upper row).

IX. Results of Indirect SIS

The method of indirect SIS is attractive as segmenting bigger-sized parts of the image (above and below the region of interest) seems to be easier than segmenting those stripes. However, it is revealed that both the upper and lower image parts cannot be directly segmented by the shallow SSN (Fig. 4), whereas using the deeper SSN twice is senseless. Indeed, the SSN trained for one epoch (see Table V) starts segmenting the empty image, so it is not trained further (Fig. 11). Results of SIS without DropOut are even worse.

TABLE V
Performance of the SSN Trained for One Epoch by Fig. 4 for the Upper Part Resulted in Badly Segmenting the Empty Image

Global accuracy	Mean accuracy	Mean IoU	Weighted IoU	Mean BF score
0.92328	0.92685	0.85748	0.85741	0.64578

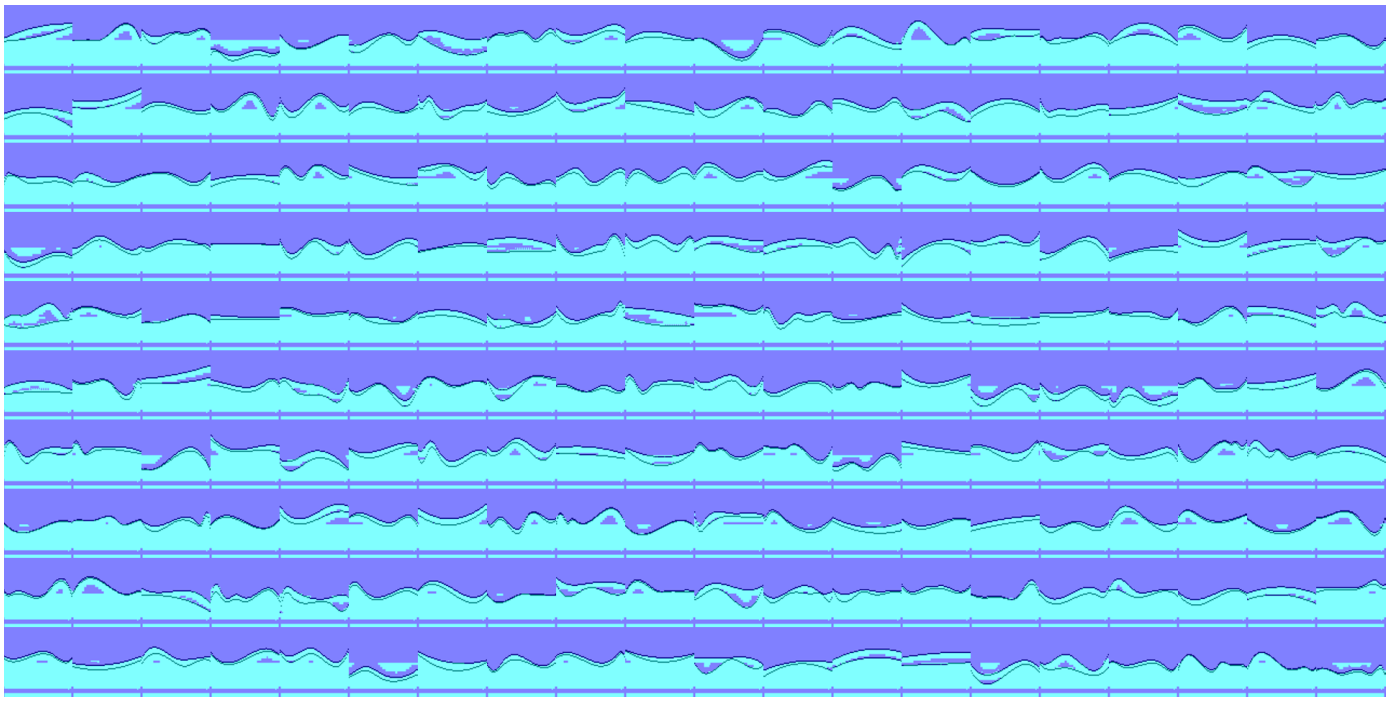

Fig. 11. SIS of the upper part. The upper border is not properly "seen". As the empty image is segmented, a straight stripe is erroneously labelled at the bottom.

SIS of the lower image part gives similar poor results. Using SSNs with the number of filters set at 96 (i.e., between 64 and 128) or less does not help. Consequently, results of indirect SIS are quite unacceptable, and this method cannot be used for segmenting curvilinear meandering regions.

X. Discussion

Why is the segmentation of upper and lower image parts so unsuccessful? After all, it might be expected to be far easier than "seeing" narrower stripes. The reason behind this perplexity exists in that segmenting bigger objects (with respect to the image size) is prone to overfitting. This was shown in [10] and [11] by examples of big convex polygons. The effect of the empty image segmentation is a key feature of such overfitting.

The subsequent question is what to do when stripes themselves are big, which can be easily generated by greater values of w_{str} in routine (1)–(18). In fact, an example of such bigger-sized regions of interest is in Fig. 2. SSNs are very hard to train on the datasets like that in Fig. 2, where the upper and lower image parts in some instances are even smaller than the region of interest (except for a few really thin stripes resembling those in Fig. 3). The solution is simple: the images with too big stripes are scaled vertically in order to artificially squeeze them to the view of much thinner stripes. Then deeper SSNs trained in the abovementioned manner are expected to perform SIS accurately, without segmenting the empty image. Once the squeezed region is segmented, the image is conversely rescaled to the original view.

If the stripes are so big that the upper and lower image parts appear themselves as really thin stripes, then they indeed can be segmented. In this case, the regions of interest cannot be named curvilinear meandering stripes. Quite the contrary, the upper and lower parts are rather stripes. Therefore, indirect SIS is not only possible here, but is just required. Then, admittedly, two SSNs (for both parts) are to be trained. Nevertheless, the solution with squeezing, segmenting, and rescaling is undoubtedly possible also for this case because it requires training of only one SSN.

XI. Conclusion

The study is a prototype model for SIS of curvilinear meandering regions. Based on the experiments carried out, it is better to fulfil SIS of smaller (or thinner, narrower) curvilinear meandering regions by the means of SSNs. If the region of interest becomes bigger, it is recommended to squeeze it in order to avoid segmenting the empty image.

The method of indirect SIS is not excluded. If the upper and lower image parts are sufficiently thin (or narrow), this method is applicable. However, indirect SIS will be practically reasonable only if the two SSNs used for segmenting the upper and lower image parts are shallower (and thus they are trained faster) than the SSN used for direct SIS. Otherwise, the method of indirect SIS with respect to curvilinear meandering regions is inapplicable.

Along with the described results and recommendations, the study is a contribution to the field of the theory of SIS. The developed toy dataset generator of curvilinear meandering regions can be used for prototyping deep SSNs before solving real-world tasks of segmenting land lines, riverbeds, highways, mountain chains, etc. (requiring only augmentation of training data, whereas augmentation for the toy datasets is needless). Another important inference (as part of the contribution) is that the mean BF score is the key accuracy indicator. The mean BF score has the least value among the other accuracy indicators, and this value should be maximized first.

References

[1] H.-J. He, C. Zheng, and D.-W. Sun, "Image Segmentation Techniques," in: *Computer Vision Technology for Food Quality Evaluation, 2nd edition*, Sun D.-W. (ed.). Academic Press, San Diego, 2016, pp. 45–63. https://doi.org/10.1016/B978-0-12-802232-0.00002-5

[2] Ç. Kaymak and A. Uçar, "A Brief Survey and an Application of Semantic Image Segmentation for Autonomous Driving," in: *Handbook of Deep Learning Applications. Smart Innovation, Systems and Technologies*, Balas V., Roy S., Sharma D., Samui P. (eds). Springer, Cham, 2019, pp. 161–198. https://doi.org/10.1007/978-3-030-11479-4_9

[3] G. Neuhold, T. Ollmann, S. R. Bulò, and P. Kontschieder, "The Mapillary Vistas dataset for semantic understanding of street scenes," *2017 IEEE International Conference on Computer Vision*, Venice, 2017, pp. 5000–5009. https://doi.org/10.1109/ICCV.2017.534

[4] J. Rogowska, "Overview and Fundamentals of Medical Image Segmentation," in: *Handbook of Medical Image Processing and Analysis, 2nd edition*, Bankman I. N. (ed.). Academic Press, San Diego, 2009, pp. 73–90. https://doi.org/10.1016/B978-012373904-9.50013-1

[5] H. Liu, J. Xu, Y. Wu, Q. Guo, B. Ibragimov, and L. Xing, "Learning deconvolutional deep neural network for high resolution medical image reconstruction," *Information Sciences*, vol. 468, pp. 142–154, 2018. https://doi.org/10.1016/j.ins.2018.08.022

[6] Larsson C. Clustering, in: *5G Networks*, Larsson C. (ed.). Academic Press, San Diego, 2018, pp. 123–141. https://doi.org/10.1016/B978-0-12-812707-0.00011-5

[7] Shapiro L. G., Stockman G. C. *Computer Vision*. Prentice-Hall, New Jersey, 2001.

[8] Sethian J. A. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4): 1591–1595, 1996. http://dx.doi.org/10.1073/pnas.93.4.1591

[9] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, iss. 12, pp. 2481–2495, 2017. https://doi.org/10.1109/TPAMI.2016.2644615

[10] V. V. Romanuke, "Generator of a toy dataset of multi-polygon monochrome images for rapidly testing and prototyping semantic image segmentation networks," *Electrical, Control and Communication Engineering*, vol. 15, no. 2, pp. 1–8, 2019. https://doi.org/10.2478/ecce-2019-0008

[11] V. V. Romanuke, "An infinitely scalable dataset of single-polygon grayscale images as a fast test platform for semantic image segmentation," *KPI Science News*, no. 1, pp. 24 –34, 2019. https://doi.org/10.20535/kpi-sn.2019.1.157259

Vadim V. Romanuke was born in 1979. He graduated from the Technological University of Podillya in 2001. The higher education was received in 2001. In 2006, he received the Degree of Candidate of Technical Sciences in Mathematical Modelling and Computational Methods. The degree of Doctor of Technical Sciences in Mathematical Modelling and Computational Methods was received in 2014. In 2016, Vadim Romanuke received the academic status of Full Professor.
He is a Professor of the Faculty of Mechanical and Electrical Engineering at the Polish Naval Academy. His current research interests concern semantic image segmentation, decision making, game theory, statistical approximation, and control engineering based on statistical correspondence.
Address for correspondence: 69 Śmidowicza Street, Gdynia, Poland, 81-127.
E-mail: romanukevadimv@gmail.com
ORCID iD: https://orcid.org/0000-0003-3543-3087