sciendo RIGA TECHNICAL UNIVERSITY

Applied Computer Systems

ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
May 2019, vol. 24, no. 1, pp. 61–68
https://doi.org/10.2478/acss-2019-0008
https://content.sciendo.com

An Improvement of the VDSR Network for Single Image Super-Resolution by Truncation and Adjustment of the Learning Rate Parameters

Vadim Romanuke*
Polish Naval Academy, Gdynia, Poland

***Abstract* – A problem of single image super-resolution is considered, where the goal is to recover one high-resolution image from one low-resolution image. Whereas this problem has been successfully solved so far by the known VDSR network, such an approach still cannot give an overall beneficial effect compared to bicubic interpolation. This is so due to the fact that the image reconstruction quality has been estimated separately by three subjective factors. Moreover, the original VDSR network consisting of 20 convolutional layers is apparently not optimal by its depth. This is why here those factors are aggregated, and the network performance is deemed by a single estimator. Then the depth is tried to be decreased (truncation) along with adjusting the learning rate drop factor. Finally, a plausible improvement of the VDSR network is confirmed. The best truncated network, performing by almost 3.2 % better than bicubic interpolation, occupies less memory space and is about 1.44 times faster than the original VDSR network for images of a medium size.**

***Keywords* – Bicubic interpolation, image similarity metrics, learning rate, single image super-resolution, truncated network, upscaled image, VDSR network.**

I. Introduction

Obtaining a high-resolution image from a low-resolution image is an open task of the modern computer vision and image analysis. Properly speaking, this is a process of increasing the resolution in order to clarify image details. The process itself is called super-resolution [1], [2]. This task issues from a lot of real-world applications retrieving information from visual data. In medicine, biometrics, microscopy, robotics, security surveillance/control [3], [4], astronomy/astrophysics [5], etc., such data are commonly insufficient to produce a desired result.

A high-resolution image can be obtained/recovered from either a few low-resolution images differing but containing the same scene, or a single low-resolution image. The first case refers to geometrical super-resolution reconstruction [4], [6]. An improved resolution image is created by fusing information from all low-resolution images. The second case is based on methods reminding interpolation of a function whose discrete values are tabulated (or presented as a mesh).

II. Analysis of the Background

Nowadays, there are promising studies on using deep convolutional networks [7], [8] to perform single image super-resolution (SISR), where the goal is to recover one high-resolution image from one low-resolution image. SISR is challenging because high-frequency image content typically cannot be recovered from the low-resolution image. Besides, SISR is an ill-posed problem because a low-resolution image can yield several possible high-resolution images. Obviously, without high-frequency information, the quality of the high-resolution image is limited [2], [3], [5].

As of 2019, the very deep super-resolution (VDSR) network is believed to be the best state-of-the-art method to perform SISR [9]. The original VDSR network consists of 20 convolutional layers, which are followed with ReLUs except for the last convolutional layer (Fig. 1). The last layer is a regression layer instead of a ReLU. The regression layer computes the mean-squared error between the residual image and network prediction. The input and output image share the same size that is achieved by padding with zeros in every convolution. VDSR employs a residual learning strategy, meaning that the network learns to estimate a residual image. The residual learning is applied by adding the input image to the output from the last convolutional layer. Only the difference between low and high resolution is learned by the network in this way. It makes sense because both images share the same low frequencies and thus do not need to be considered in the training process. As a result, such a residual network converges much faster [10]. It takes between 10 to 40 epochs to achieve the top performance [9].

```
 1 'InputLayer'       Image Input   41x41x1 images
 2 'Conv1' 64 3x3x1 convolutions with stride [1  1] and padding [1  1  1  1]
 3 'ReLU1' ReLU
 4 'Conv2' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
 5 'ReLU2' ReLU
 6 'Conv3' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
 7 'ReLU3' ReLU
 8 'Conv4' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
 9 'ReLU4' ReLU
10 'Conv5' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
11 'ReLU5' ReLU
12 'Conv6' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
13 'ReLU6' ReLU
14 'Conv7' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
15 'ReLU7' ReLU
16 'Conv8' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
17 'ReLU8' ReLU
18 'Conv9' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
19 'ReLU9' ReLU
20 'Conv10' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
21 'ReLU10' ReLU
22 'Conv11' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
23 'ReLU11' ReLU
24 'Conv12' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
25 'ReLU12' ReLU
26 'Conv13' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
27 'ReLU13' ReLU
28 'Conv14' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
29 'ReLU14' ReLU
30 'Conv15' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
31 'ReLU15' ReLU
32 'Conv16' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
33 'ReLU16' ReLU
34 'Conv17' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
35 'ReLU17' ReLU
36 'Conv18' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
37 'ReLU18' ReLU
38 'Conv19' 64 3x3x64 convolutions with stride [1  1] and padding [1  1  1  1]
39 'ReLU19' ReLU
40 'Conv20' 1 3x3x64 convolution with stride [1  1] and padding [1  1  1  1]
41 'FinalRegressionLayer' Regression Output mean-squared-error with response 'residuals'
```

Fig. 1. The architecture of the VDSR network in Matlab. Apart from the first and last layers, this architecture has 18 identical convolutional layers [9].

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

In the VDSR network, the filter size of each convolution, except the first one, is $3\times3\times64$. Due to those 20 convolutional layers, the receptive field (i.e., information used for reconstruction) of the network is therefore 41×41 pixels. Owing to this, the training data is decomposed into patches with size 41×41. That helps in gaining speed and reducing the size of the network. Besides, data augmentation via rotation and flipping is used for training to improve generalization. Owing to combining several specified scales into one big dataset, the VDSR network is trained straightforwardly as a multi-scale model.

The VDSR performance is compared to classic naive approaches of interpolation: bilinear [11], bicubic [12], and nearest-neighbour [13]. The nearest neighbour method simply predicts the pixel values from the value of the nearest neighbour pixel. This method is pretty rough because the reconstructed image is too pixelized compared to bilinear interpolation, which considers 4 surrounding pixels to predict new pixel values. Bicubic interpolation considering 16 surrounding pixels to predict new pixel values is a bit better than bilinear interpolation, but the improvement is not always easy observable (Fig. 2).

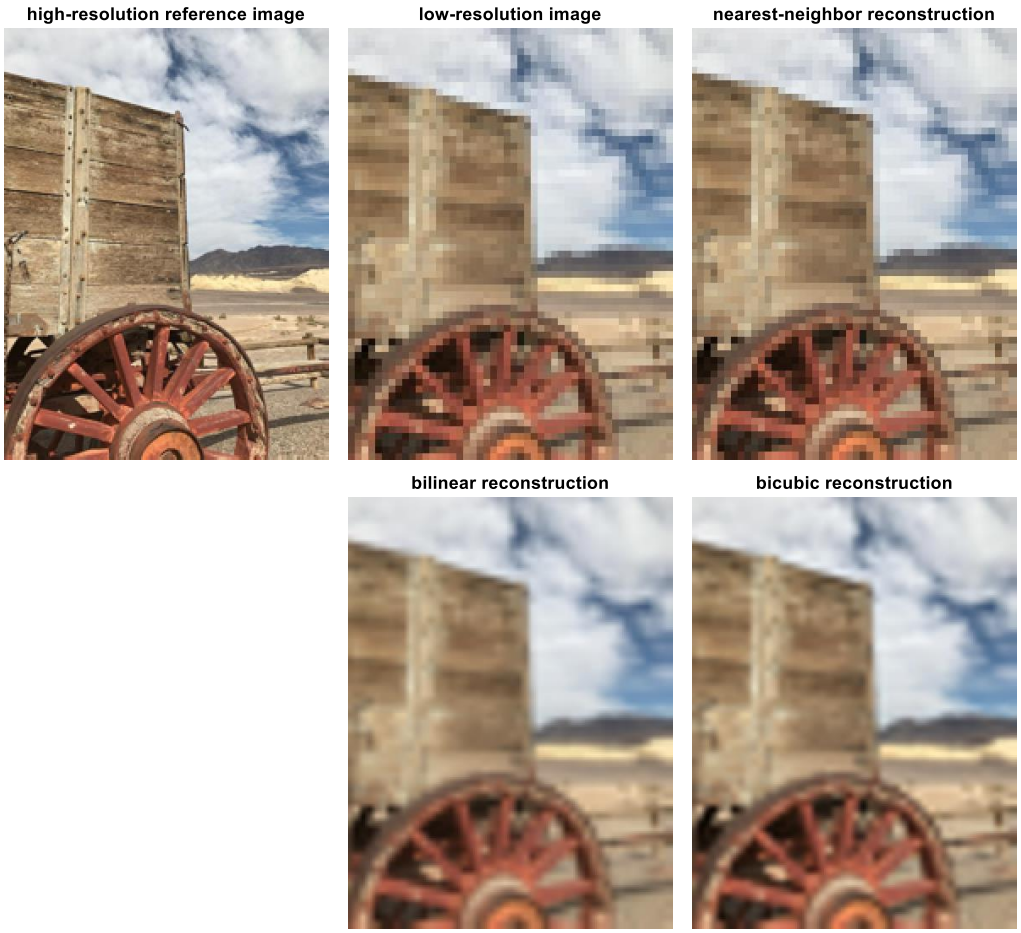

Fig. 2. A comparative example of image reconstruction by 4 times upscaling: pixelization versus blurring. Bilinear and bicubic reconstructions blur the image.

The quality of the upscaling method is described by a metric to measure the similarity between the predicted (or upscaled) image and the ground truth image. There are several commonly used metrics. Traditionally, the peak signal-to-noise ratio (PSNR) defines the similarity between two images by calculating a ratio with the mean-square-error (of the respective pixels of the two images) in the denominator. Greater PNSR values generally indicate better reconstruction quality [9].

Another metric is the structural similarity index (SSIM), which assesses the visual impact of luminance, contrast, and structure of an upscaled image against a reference image [14]. The SSIM index is calculated on various windows of an image. The closer the SSIM value is to 1, the better the upscaled image agrees with the reference image.

To measure perceptual image quality, the naturalness image quality evaluator (NIQE) is used [15]. NIQE operates on a feature set based on natural scene statistics modelled as multi-dimensional Gaussian distributions. Lower values of NIQE reflect better perceptual quality of the image.

Despite the apparent advantages of VDSR, the PSNR, SSIM index, and NIQE for the VDSR network are not always better than those for bicubic interpolation. Deeper VDSR networks having up to 30 convolutional layers do not outperform the original VDSR much [16]. Indeed, the VDSR network seems too straightforward by its equal-sized convolutions. Such straightforwardness has been not proved to be optimal. All the more, many studies reported that it was more efficient to increase the number of convolutions throughout the network

[7], [10], [17]. Besides, the learning rate drop factor (LRDF), suggested along with the original network in [9], decreases the initial learning rate of 0.1 by 10 every 20 epochs. This also appears weakly substantiated. Therefore, it seems that either improving the PSNR, SSIM index, and NIQE by the same operability, or reducing the size of the VDSR network is plausible.

III. Goal and Steps to Achieve It

Due to the seeming plausibility to improve the VDSR network, the goal is to confirm or refute it. For doing this, the network first will be truncated by decreasing the number of convolutional layers down off 20. The network performance is to be estimated by averaging the PSNR, SSIM index, and NIQE against those ones for bicubic interpolation. Then, the number of convolutions throughout the network will be increased up off 64. A few laws of the increment are to be tried. For all those experiments, the same benchmark dataset will be used by attempting to vary the learning rate and LRDF. On this dataset, a benchmark VDSR network will be reproduced by the same training parameters and scale factors (2, 3, 4) as they were suggested for the original VDSR in [9]. Thus, the performances of new networks are to be compared to the performance of the benchmark (original) VDSR network.

IV. The Network Performance Estimator

Denote the PSNR, SSIM index, and NIQE for the VDSR network and bicubic interpolation by $t_{\text{VDSR}}(i,s)$, $n_{\text{VDSR}}(i,s)$, $h_{\text{VDSR}}(i,s)$, $t_{\text{bicubic}}(i,s)$, $n_{\text{bicubic}}(i,s)$, $h_{\text{bicubic}}(i,s)$, respectively, where i is the index/tag of an image, and s is a scale factor by $s \in \{2,3,4\}$. Then a complex gain of reconstructing the i-th image by s times upscaling it is

$$\mathbf{G}(i,s)=\left[\frac{t_{\text{VDSR}}(i,s)}{t_{\text{bicubic}}(i,s)} \quad \frac{n_{\text{VDSR}}(i,s)}{n_{\text{bicubic}}(i,s)} \quad \frac{h_{\text{bicubic}}(i,s)}{h_{\text{VDSR}}(i,s)}\right]. \quad (1)$$

Obviously, if all the three elements of vector (1) are greater than 1, then VDSR is absolutely better than bicubic interpolation for the i-th image s times upscaled. Sometimes, however, VDSR loses to bicubic interpolation by at least one of those elements. Thus, it is admissible to use a generalized gain for the i-th image:

$$g(i)=\frac{1}{9}\sum_{s=2}^{4}\left(\frac{t_{\text{VDSR}}(i,s)}{t_{\text{bicubic}}(i,s)}+\frac{n_{\text{VDSR}}(i,s)}{n_{\text{bicubic}}(i,s)}+\frac{h_{\text{bicubic}}(i,s)}{h_{\text{VDSR}}(i,s)}\right). \quad (2)$$

The VDSR performance gain for a whole test set of M images is

$$\tilde{g}=\frac{1}{M}\sum_{i=1}^{M} g(i). \quad (3)$$

Formulae (2) and (3) both define the network performance estimator, whereas gain (3) allows comparing networks among themselves by just a point estimate. By using gain (2), it is possible to find poorly reconstructible images within a test set.

V. The Benchmark Dataset

Unlike datasets for other image analysis tasks deep learning deals with, a dataset for training to perform SISR cannot contain tiny images. A remarkable specificity of an SISR dataset is that it can consist of variously-sized images whose scenes may not be related. This allows considering a one big dataset as a benchmark. Training and testing results on other such benchmarks will not differ significantly.

Thus, a dataset consisting of 616 colour images is chosen for training (Fig. 3). A test set of 67 colour images is shown in Fig. 4 (both sets are available at informatik.rwth-aachen.de).

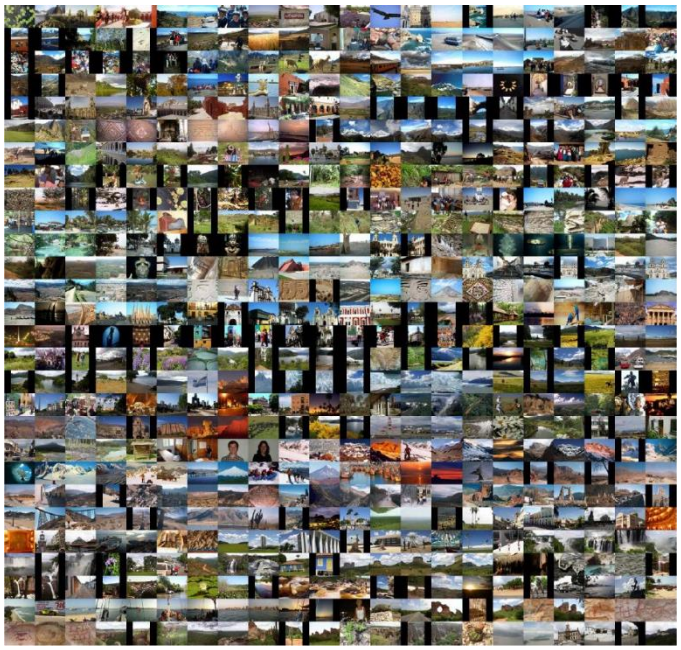

Fig. 3. A dataset for training VDSR networks. It consists of 500 360×480 images, and 116 480×360 images. The image scenes are very heterogeneous.

Fig. 4. A set of 67 images for testing VDSR networks. It consists of 51 320×480 images, and 16 480×320 images. The volume of this set is about 11 % of the volume of the dataset for training, which is usual in practice.

VI. Truncation of the Original VDSR Network

Authors of the original VDSR network argued [9] that an increase in the network depth (the number of convolutional layers) improved the performance rapidly. As a confirmation, they showed three plots of the depth versus performance (Fig. 5). However, those plots showed only PSNR, so this estimation of performance was not complete. The other two metrics were ignored, whereas they might be contradictory to the PSNRs in Fig. 5. Besides, some strange peaks and valleys may be artifacts of insufficient volume of testing. What is the most important, there is no comparison to bicubic interpolation, which is much faster than deep learning.

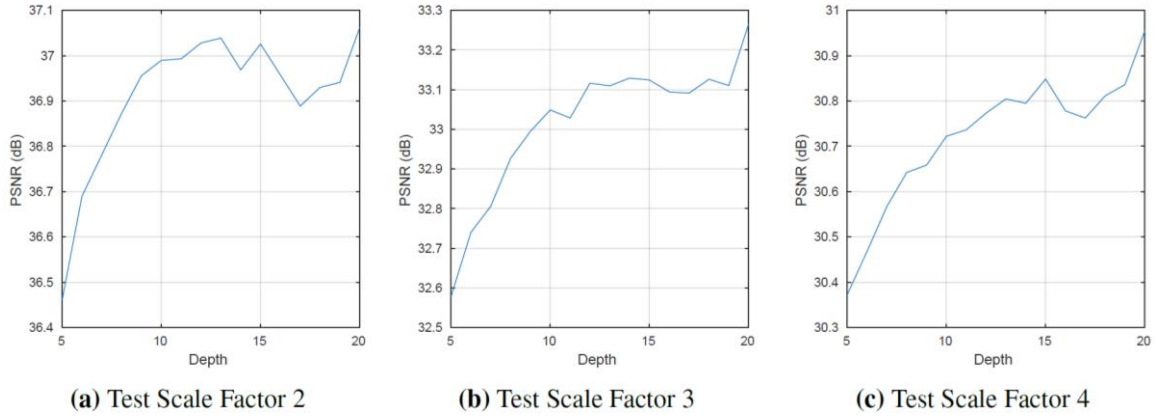

Fig. 5. A screenshot of Fig. 3 in paper [9], wherein the original VDSR was presented. The PSNR is shown versus the network depth. A peak at 15 convolutional layers is easily observed for the minimal and maximal scale factors. Unexpectedly, there is also a valley at 17 convolutional layers, although both peak and valley for $s = 3$ are less apparent. After all, the PSNR polyline for the medium upscaling is the least expressive. The most expressive is that for the minimal upscaling.

It should be recalled that due to the lesser number of convolutional layers (the shallower depth), the receptive field of the network is smaller. As a result, the training data is decomposed into smaller patches. This particularly explains why building a VDSR network of 10 convolutional layers or less is ineffective. The PSNR maximum at 13 convolutional layers (without considering the maximum at 20) is nonetheless unconvincing. Therefore, an impact of smaller depths (which can be thought of as a truncation with respect to the network with 20 convolutional layers by Fig. 1) should be revised.

Figure 6 shows gain (3) versus the depth for VDSR networks trained on the dataset in Fig. 3 and tested on the dataset in Fig. 4. It is well seen that the factual global maximum is at 14 convolutional layers (let the respective network be called VDSR-14). Despite the peak at 18 (network VDSR-18) is really close up to that at 14, operability of VDSR-14 is better owing to its shallower depth (here, in fact, 4 redundant convolutional layers are thrown away). Moreover, VDSR-14 occupies 1.58 MB against 2.11 MB of VDSR-18. Nonetheless, the gain of VDSR-20 (a 10-epoched basis for the original VDSR network occupying 2.37 MB) loses to that of both VDSR-14 and VDSR-18.

The global minimum of the polyline in Fig. 6 implies that VDSR-16 not only loses to every other network, but also directly loses to bicubic interpolation. Although the polyline resembles the one from the left in Fig. 5, the peaks and valleys do not coincide. Figure 7 showing gain (2) for all the 67 images confirms that the top gains (peaks at VDSR-14 and VDSR-18) and the depth "fails" (valleys at VDSR-16 and VDSR-19) in Fig. 6 are not a result of randomness.

It should be noted that each of those 11 networks loses to bicubic interpolation on reconstructing specific images. The loss of VDSR-14 is the least, constituting 11 images (Fig. 8). Compared to bicubic interpolation, SISR is performed worst by VDSR-16, which "spoiled" a half of the test images.

Hence, truncating VDSR-20 to VDSR-14 has two beneficial effects: gain (3) is improved by 0.74 % (from 1.0113 to 1.0188), and the size of the network is reduced by one third (from 2.37 MB to 1.58 MB). Obviously, the gain improvement is very subtle, but it is statistically consistent. Next, we will try to improve the gain (which being greater than 1 is equivalent to performance) by increasing the number of convolutions throughout the network up off 64.

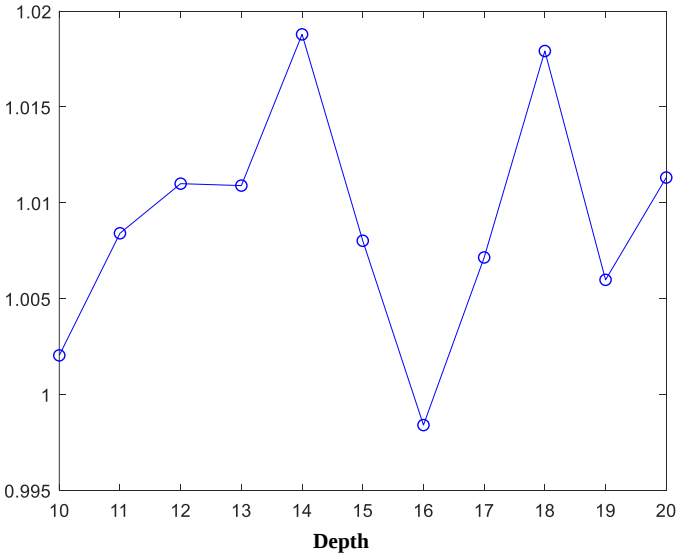

Fig. 6. Gain (3) versus the number of convolutional layers in the VDSR network. Each network is trained for 10 epochs, which allow obtaining nearly the top performance. The gain is averaged over those three scale factors. Unlike it is in Fig. 5, the factual global maximum now is at 14 convolutional layers. The peak at 18 is quite close to that maximal gain. The valley at 16 (the gain is less than 1) does not coincide with the analogous valley in Fig. 5.

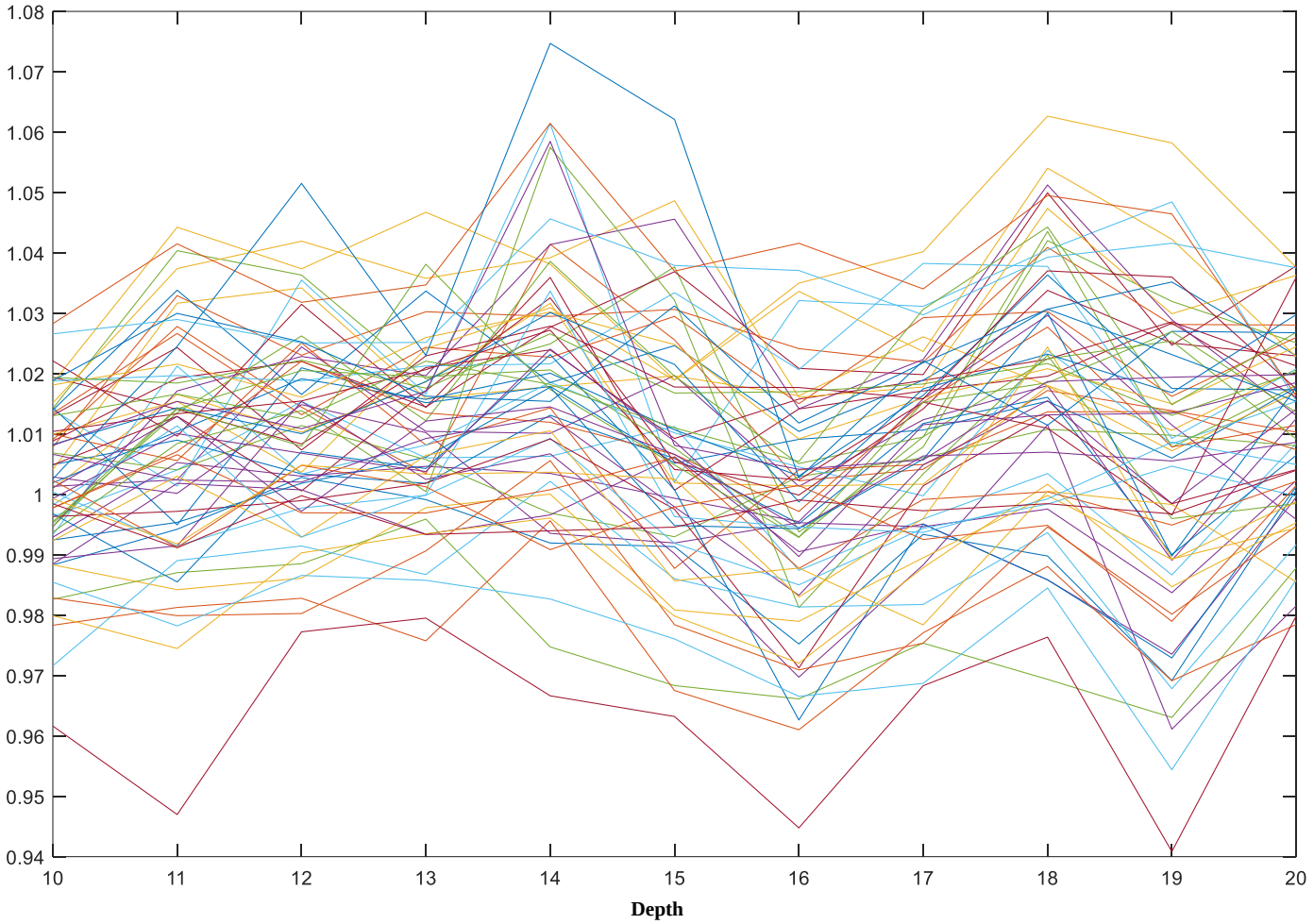

Fig. 7. Gain (2) for every image in Fig. 4 versus the number of convolutional layers in the VDSR network. The gains at VDSR-14 appear the best, although the bunch of gains at VDSR-13 is narrower. The widest bunch of gains is at VDSR-19. The gains at VDSR-14 are badly scattered too, but the scattering at VDSR-19 is "directed" downwards. Anyway, these bunches of gains confirm that VDSR-20 is pretty far from the optimal deep learning architecture for performing SISR.

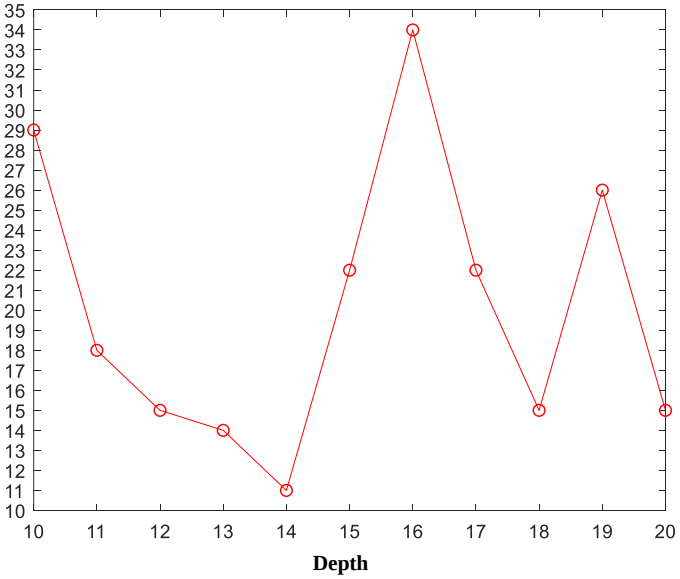

Fig. 8. The number of images reconstructed better by bicubic interpolation versus the number of convolutional layers in the VDSR network. The advantage is meant by the three metrics considered simultaneously to estimate the image reconstruction quality. The plotted polyline looks like turned upside down compared to Fig. 6, except for the right side "tails" of these polylines. Thus, the global minimum is at 14, and the global maximum is at 16.

VII. Convolution Increment

First of all, we should try a slight increment without touching training parameters of the original VDSR network (remember it differs from VDSR-20 in being trained 8 times longer). For this purpose, let every next convolutional layer, starting from the second one, have by 2 convolutions more than the previous layer has. By the same depth of the network (20 convolutional layers), the convolutions are 64, 66, 68, 70, ..., 96, 98, 100 (instead of 64, 64, 64, ..., 64, 64, 64), where the last convolutional layer is still of a single 3×3×100 convolution (this is why the convolution in the 20-th layer is not listed). However, the corresponding experiments show that such an architecture does not outperform the original one in Fig. 1. The truncation of such slightly incremented convolutions gives no effect as it is for VDSR-14. Decreasing the learning rate to 0.01 or/and setting LRDF to 0.95 for every 10 epochs (unlike the original LRDF of 0.1 for every 20 epochs) may even worsen the performance.

Meanwhile, it would be interesting to see what happens when the number of the starting convolutions is less than 64. Thus, trying this number at 8, 16, 32 with the subsequent increment of convolutions by

$$c_d = c_{d-1} + 2,\ d = \overline{2,19},\ c_1 \in \{8, 16, 32\}, \quad (4)$$

where d is the index of the convolutional layer with c_d convolutions, has no effect. Training for longer periods does not help at all. Therefore, a local conclusion here is that 64 convolutions in the start ($c_1 = 64$) are close to optimal.

Since the slight progression

$$c_d = c_{d-1} + 2,\ d = \overline{2,D},\ D = \overline{10,19},\ c_1 = 64, \quad (5)$$

does not fit for SISR, nor fits scheme (4), it is normal to try more "aggressive" increments. One of such incremental schemes is

$$c_d = c_{d-1} + 4 + 2(d-2),\ d = \overline{2,19},\ c_1 = 64, \quad (6)$$

which still gives us adjacent convolutional layers with different convolutions. However, the number of convolutions spans here from 64 to 442 (64, 68, 74, 82, 92, ..., 334, 368, 404, 442) that makes such a network occupy almost 30 MB. Moreover, the network performance is not improved, so scheme (6) does not fit as well. Other incremental schemes, in which every next convolutional layer always has a greater number of convolutions (and this difference increases from layer to layer), do not work either.

In spite of the negative result for convolution increment by (6), the performance for networks with the "aggressive" increments is nonetheless increasing as LRDF is defined greater (up to 0.95) for every 10 epochs. Some networks occasionally achieve roughly the same performance as the original network has. Eventually, another incremental scheme, which is

$$c_1 = 64,\ c_{2d} = c_{2d-1},\ c_{2d+1} = c_{2d} + 4 + 4(d-1),\ d = \overline{1,7} \quad (7)$$

allows outperforming both the VDSR-14 and original VDSR network: after 100 epochs of training, gain (3) now is 1.0316 against those 1.0188 (Fig. 6; trained for 10 epochs) and 1.0182 (originally, the VDSR network was trained for 80 epochs [9]), respectively. Nevertheless, the network whose architecture is built by scheme (7) occupies 4.91 MB, which is more than twice "heavier" than VDSR-20. For further references, let this network be called VDSR-16p4 inasmuch it has 16 convolutional layers, and figure "4" is a significant part of progression (letter "p") in scheme (7).

An interesting fact is that if the network by the original architecture (Fig. 1) is trained identically to VDSR-16p4 (i.e., throughout 100 epochs with the learning rate of 0.1 and LRDF of 0.95 for every 10 epochs), then it achieves almost the same performance (Fig. 9). For further references, let this network be called VDSR-20/0.95.

It is well seen in Fig. 9 that both polylines for VDSR-16p4 and VDSR-20/0.95 increase similarly. The best gain of VDSR-20/0.95 is 1.0302, which is statistically very close to the best gain of VDSR-16p4 (1.0316). Therefore, VDSR-16p4 and VDSR-20/0.95 have the same performance. In this way, a version of VDSR with 14 convolutional layers (the VDSR-14 architecture) should be tried similarly to VDSR-20/0.95 (dropping the learning rate more frequently but far less intensively). Eventually, VDSR-14/0.95 achieves the performance of the VDSR-16p4 and VDSR-20/0.95 (Fig. 10), although it takes more epochs.

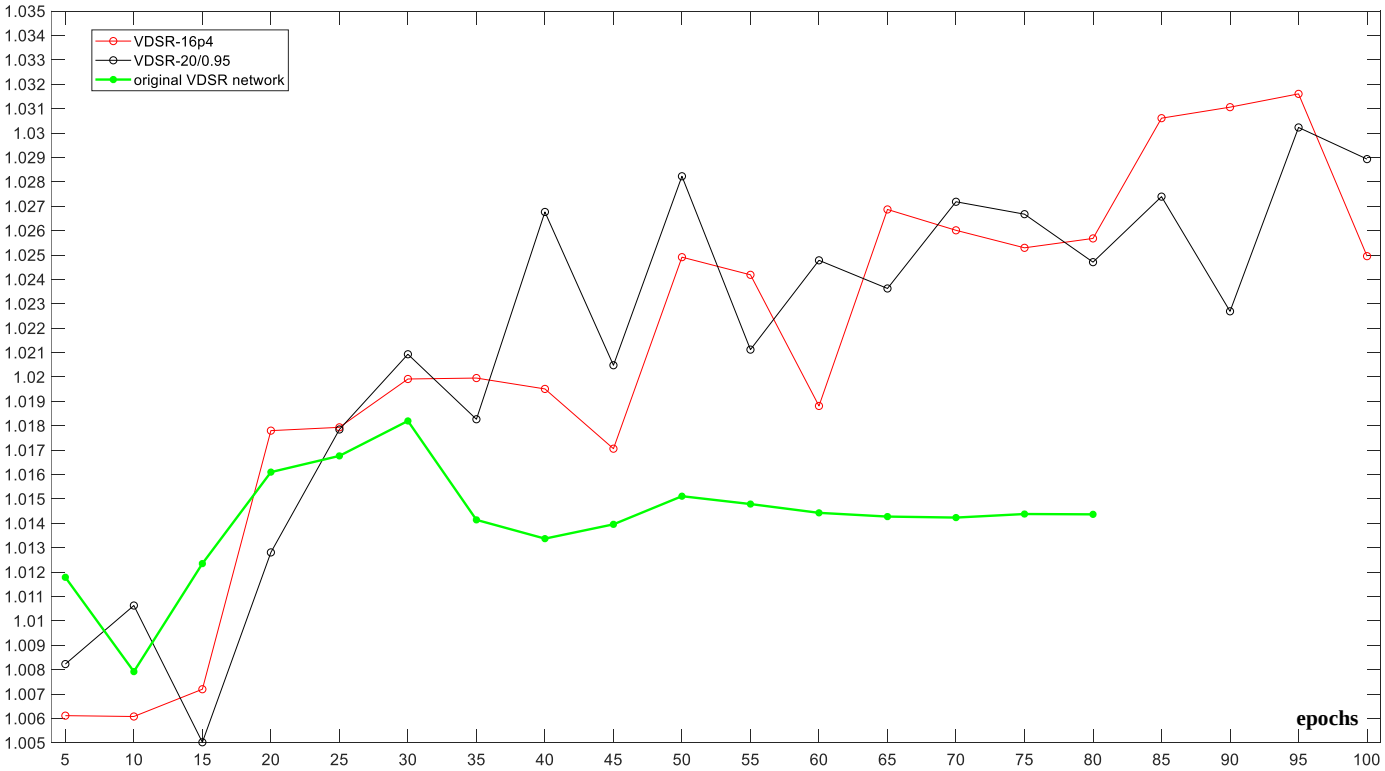

Fig. 9. Gains (3) for VDSR-16p4 and VDSR-20/0.95 against the original VDSR network. At the starting 20 epochs the original VDSR network acquires some advantage over VDSR-16p4 and VDSR-20/0.95. After the 30-th epoch, the performance of VDSR-16p4 and VDSR-20/0.95 is significantly better than that of the original VDSR network. The latter after the 30-th epoch worsens and since the 50-th epoch it stays almost at the same rate (even decreasing a little bit).

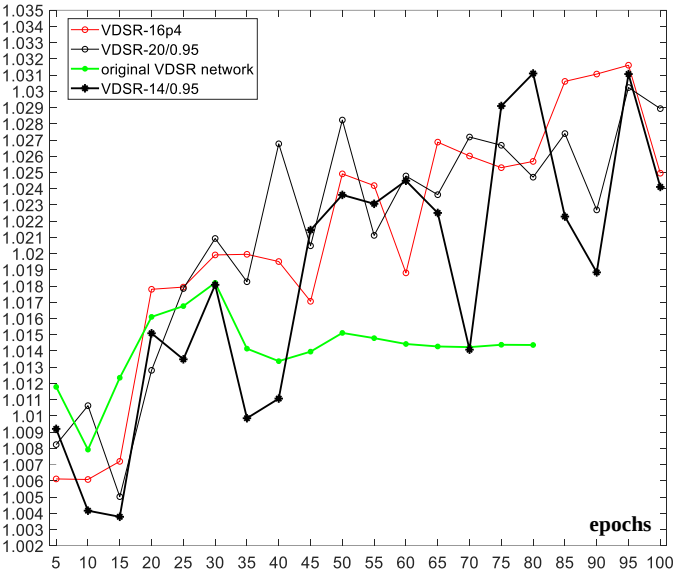

Fig. 10. Gain (3) for VDSR-14/0.95 that achieves the top rate of VDSR-16p4 and VDSR-20/0.95. During the first 40 epochs, the performance of VDSR-14/0.95 is worse than that of the original VDSR. Then the gain grows similarly to VDSR-16p4 and VDSR-20/0.95 achieving its top of 1.0311 (twice). Unlike the gain for VDSR-16p4 and VDSR-20/0.95, the gain for VDSR-14/0.95 is more unstable: there are four distinct valleys that are badly deeper than those for VDSR-16p4 and VDSR-20/0.95.

It must be noted that while VDSR-14 is better that VDSR-20 (by the gain in Fig. 6 and losses to bicubic interpolation in Fig. 8), it is so just after 10 epochs. Having varied LRDF and reduced the drop period twice, VDSR-14/0.95 acquires its advantage over VDSR-20/0.95 slower. The "tardiness" of the advantage of VDSR-16p4 and VDSR-20/0.95 can be seen during the first 15 epochs as well.

VIII. Discussion

Considering the performance/gain alone, truncation of the original VDSR network has a weak overall beneficial effect. Adding the network's "weight", where a "lighter" network is faster operable than a "heavier" one, the truncation benefit is far more substantial. For example, in reconstructing images of a medium size (320×480) for SISR, VDSR-14/0.95 is about 1.44 times faster than the original VDSR network. In its turn, VDSR-16p4 is about 2.39 times slower than the original VDSR network, so application of VDSR-16p4 is only reasonable on very fast-operating computational systems.

Accuracy of SISR is too dependent on subjective factors. It is plausible that for images of definite categories averaging the PSNR, SSIM index, and NIQE by no weighting them, using formulae (1)–(3), may be biased. Visual perception (just by human eye) plays a very important role as well, sometimes contradicting with gains (1) and (2) for a fragment of an image. For example, the result of SISR shown in Fig. 11 is apparently better for VDSR-14/0.95, but in this case PSNR and NIQE are better for bicubic interpolation. The SSIM index for estimating SISR of such images should be perhaps taken with a greater weight.

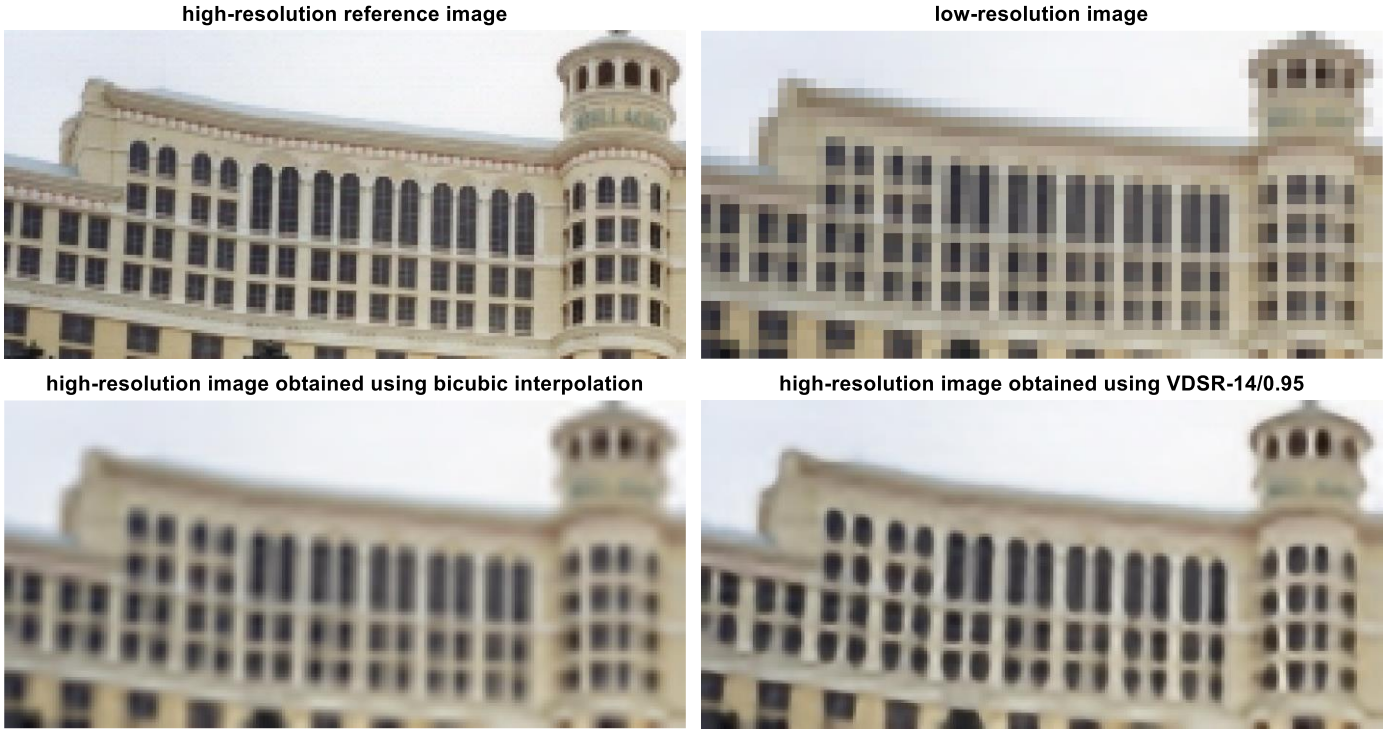

Fig. 11. A comparative result of SISR for a 101×211 fragment of an image from the test set (Fig. 4). The fragment is 3 times downscaled. The outcome by VDSR-14/0.95 appears better than that by bicubic interpolation. Amazingly enough, gain (2) here is less than 1. It is another example of the metrics subjectivity.

The crucial obstacle with VDSR networks truncated to 16 or 14 convolutional layers is their instability while they are trained. To cope with it, the number of epochs must be increased. Therefore, the faster network "pays the price", which is a longer training.

As to VDSR-14/0.95, LRDF set to 0.95 may be not optimal. The same concerns the initial learning rate of 0.1 and the drop period (re-set to 10). It is hard to claim which of these three the most sensitive is. They are not embarrassingly parallel – somehow or other, additional optimization of those training options is not believed to lead to significant improvement of the VDSR-14/0.95 performance.

Finally, we have seen that principle "the deeper, the better" does not work for VDSR. The depth of efficient VDSR networks has its limit. In general, this limit is 20. However, for some special categories of images (or their fragments) the limit may vary violating the results in Fig. 6 or/and Fig. 8 (just like the contradictory result in Fig. 11 violates the rule of the greater-than-1 gain).

IX. Conclusion

Based on the experiments carried out, the improvement of the VDSR network is confirmed. There are four networks (VDSR-14, VDSR-14/0.95, VDSR-20/0.95, VDSR-16p4, which are sorted in ascending order of the memory space occupied) whose relative performance is slightly better than that of the original VDSR network. The relative performance is estimated by averaging the PSNR, SSIM index, and NIQE, using formulae (1)–(3), against those ones for bicubic interpolation. The ratios in (1) and this averaging allows catching a real advantage of VDSR for SISR, rather than just improving the performance of VDSR networks without concerning other techniques for SISR (like bicubic interpolation), which may occasionally happen to be more accurate.

At the moment, the best VDSR network performs by almost 3.2 % better than bicubic interpolation. Each of VDSR-14/0.95, VDSR-20/0.95, and VDSR-16p4 networks outperforms it by 3 % at least. These networks are practically equivalent by the performance. Despite VDSR-14/0.95 is trained faster, the performance (along with the advantage over bicubic interpolation) of VDSR-20/0.95 and VDSR-16p4 is more stable as training progresses. VDSR-14 is trained the fastest and still has a little advantage. Being trained on a heterogeneous dataset (like that in Fig. 3; for bigger datasets, training should be for a few hundred epochs), those networks acquire good generalization properties for SISR and thus can be regarded as a close-to-universal SISR technique.

References

[1] J. Salvador, *Example-Based Super Resolution*. Academic Press, 2017, 162 p. https://doi.org/10.1016/C2015-0-06719-3

[2] V. Bannore, *Iterative-Interpolation Super-Resolution Image Reconstruction. A Computationally Efficient Technique*. Springer-Verlag Berlin Heidelberg, 2009, 113 p. https://doi.org/10.1007/978-3-642-00385-1

[3] A. Small and S. Stahlheber, "The Role of Image Analysis Algorithms in Super-resolution Localization Microscopy," in: *Fluorescence Microscopy*, Cornea A., Conn P. M. (eds.). Academic Press, 2014, pp. 227–242. https://doi.org/10.1016/B978-0-12-409513-7.00016-6

[4] M. Amiri, A. Ahmadyfard, and V. Abolghasemi, "A fast video super resolution for facial image," *Signal Processing: Image Communication*, vol. 70, 2019, pp. 259–270. https://doi.org/10.1016/j.image.2018.10.008

[5] R. Guo, X. Shi, Y. Zhu, and T. Yu, "Super-resolution reconstruction of astronomical images using time-scale adaptive normalized convolution," *Chinese Journal of Aeronautics*, vol. 31, no. 8, pp. 1752–1763, 2018. https://doi.org/10.1016/j.cja.2018.06.002

[6] I. Haq and A. A. Mudassar, "Geometric super-resolution using negative rect mask," *Optik*, vol. 168, pp. 323–341, 2018. https://doi.org/10.1016/j.ijleo.2018.04.033

[7] K. Hayat, "Multimedia super-resolution via deep learning: A survey," *Digital Signal Processing*, vol. 81, pp. 198–217, 2018. https://doi.org/10.1016/j.dsp.2018.07.005

[8] J. Lu, W. Hu, and Y. Sun, "A deep learning method for image super-resolution based on geometric similarity," *Signal Processing: Image Communication*, vol. 70, pp. 210–219, 2019. https://doi.org/10.1016/j.image.2018.10.003

[9] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1646–1654, 2016. https://doi.org/10.1109/CVPR.2016.182

[10] W. Xie, Y. Li, and X. Jia, "Deep convolutional networks with residual learning for accurate spectral-spatial denoising," *Neurocomputing*, vol. 312, pp. 372–381, 2018. https://doi.org/10.1016/j.neucom.2018.05.115

[11] R. Kress, "Interpolation," in: *Numerical Analysis*, Kress R. (ed.). Springer, 1998, pp. 151–188. https://doi.org/10.1007/978-1-4612-0599-9_8

[12] F. Aràndiga, "A nonlinear algorithm for monotone piecewise bicubic interpolation," *Applied Mathematics and Computation*, vol. 272, no. 1, pp. 100–113, 2016. https://doi.org/10.1016/j.amc.2015.08.027

[13] A. M. Bayen and T. Siauw, "Interpolation," in: *An Introduction to MATLAB® Programming and Numerical Methods for Engineers*, Bayen A. M., Siauw T. (eds.). Academic Press, 2015, pp. 211–223. https://doi.org/10.1016/B978-0-12-420228-3.00014-2

[14] D. Brunet, J. Vass, E. R. Vrscay, and Z. Wang, "Geodesics of the structural similarity index," *Applied Mathematics Letters*, vol. 25, no. 11, pp. 1921–1925, 2012. https://doi.org/10.1016/j.aml.2012.03.001

[15] A. Mittal, R. Soundararajan, and A. C. Bovik, "Making a "completely blind" image quality analyzer," *IEEE Signal Processing Letters*, vol. 20, no. 3, pp. 209–212, 2013. https://doi.org/10.1109/LSP.2012.2227726

[16] L. Zhou, Z. Wang, S. Wang, and Y. Luo, "Coarse-to-Fine Image Super-Resolution Using Convolutional Neural Networks," in: *MultiMedia Modeling. MMM 2018. Lecture Notes in Computer Science*, Schoeffmann K. et al. (eds.). Springer, 2018, pp. 73–81. https://doi.org/10.1007/978-3-319-73600-6_7

[17] V. V. Romanuke, "Appropriateness of numbers of receptive fields in convolutional neural networks based on classifying CIFAR-10 and EEACL26 datasets," *Electrical, Control and Communication Engineering*, vol. 14, no. 2, pp. 157–163, 2018. https://doi.org/10.2478/ecce-2018-0019

Vadim 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 Navigation and Naval Weapons at the Polish Naval Academy. His current research interests concern decision making, game theory, statistical approximation, and control engineering based on statistical correspondence. He has written 333 scientific articles, one monograph, one tutorial, and three methodical guidelines in Functional Analysis, Development of Master Theses in Mathematical and Computer Modelling, Conflict-Controlled Systems. Before January 2018, Vadim Romanuke was the scientific supervisor of a Ukrainian budget grant work concerning minimization of water heat transfer and consumption. He also directs a branch of fitting statistical approximators at the Center of Parallel Computations in Khmelnitskiy, Ukraine.

Address for correspondence: Śmidowicza Str. 69, Gdynia, Poland, 81-127.

E-mail: romanukevadimv@gmail.com

ORCID iD: https://orcid.org/0000-0003-3543-3087