DE GRUYTER OPEN
RIGA TECHNICAL UNIVERSITY
Information Technology and Management Science
ISSN 2255-9094 (online)
ISSN 2255-9086 (print)
December 2017, vol. 20, pp. 12–19
doi: 10.1515/itms-2017-0002
https://www.degruyter.com/view/j/itms

Appropriate Number of Standard 2×2 Max Pooling Layers and Their Allocation in Convolutional Neural Networks for Diverse and Heterogeneous Datasets

Vadim V. Romanuke
Polish Naval Academy, Poland

***Abstract* – A problem of appropriately allocating pooling layers in convolutional neural networks is considered. The consideration is based on CIFAR-10, NORB, and EEACL26 datasets for preventing "overfitting" in a solution of the problem. For highly accurate image recognition within these datasets, the networks are used with the max pooling operation. The most common form of such operation, which is a 2×2 pooling layer, is applied with a stride of 2 without padding after convolutional layers. Based on performance against a series of the network architectures, a rule for the best allocation of max pooling layers is formulated. The rule is to insert a few pooling layers after the starting convolutional layers and to insert a one pooling layer after the last but one convolutional layer ("11...100...010"). For much simpler datasets, the best allocation is "11...100...0".**

***Keywords* – Convolutional neural network, max pooling layer.**

I. Introduction

Pooling is an important concept of convolutional neural networks (CNNs) purposed for non-linear down-sampling. It is common to periodically insert a pooling layer in-between successive convolutional layers (ConvLs) in a CNN architecture. The pooling layer progressively reduces the spatial size of the representation. Thus, it reduces the number of parameters and the amount of computation in the network [1], [2]. Pooling is also aimed at controlling overfitting [3].

Reducing computations by pooling, however, increases the total training time. It is obvious that too much pooling may be destructive. An important question is how many pooling layers are needed for a given image classification problem. Number of pooling layers cannot be greater than that of ConvLs. Whether each ConvL should be followed by pooling layers or not, is another topic open for discussion.

II. Background

The pooling layer operates independently of a depth slice of the input and resizes it spatially. Operation of either maximisation or averaging is used. The most common form is a 2×2 pooling layer applied with a stride of 2 without padding (standard form). This downsamples every depth slice in the input by 2 along both width and height, discarding 75 % of the activations. If there is a maximisation operation, then a maximum over 4 numbers which represent a 2×2 region in some depth slice is taken. The depth dimension remains unchanged [4].

Average pooling has recently fallen out of favour compared to the max pooling operation, which has been shown to work better in practice (e.g., see [5]). It is not common to use zero-padding. In general, the max pooling layer (MPL) requires two hyperparameters – the pooling spatial extent F and the stride S. MPL accepts a volume of size $W_1 \times H_1 \times D_1$ and produces a volume of size $W_2 \times H_2 \times D_2$, where

$$W_2 = \frac{W_1 - F}{S} + 1, \quad H_2 = \frac{H_1 - F}{S} + 1, \quad D_2 = D_1.$$

There are only two commonly seen variations of MPL, both having the same stride, which is $S = 2$. The most spread version of MPL mentioned above (standard) has $F = 2$ [4]. The other one has $F = 3$ which along with $S = 2$ performs an overlapping pooling [6]. Fractional pooling includes them [7].

There is an idea of constructing CNNs without pooling [8]. The size of the representation is suggested to be reduced using larger stride in ConvLs once in a while. Some novel CNN architectures tend to have a very few to no pooling layers.

III. Goal

The goal of the present paper is to formulate a rule for the best allocation of standard 2×2 MPLs. To achieve the goal, the following tasks are to be fulfilled:

1. To substantiate a benchmark image classification problem (or a group of such problems).
2. To generate a series of all admissible versions for MPL allocation, depending on the image size and number of ConvLs.
3. To evaluate the CNN performance depending on each element of the series.
4. To extract a version of MPL allocation that maximises the performance, for each image size, if possible.
5. To formulate a rule for the best MPL allocation.

IV. CIFAR-10, NORB, EEACL26 Datasets for Benchmark

To evaluate performance correctly and consistently, the benchmark image classification problem should be spacious and blanket. An example of such an image classification problem is the ImageNet dataset [9]. However, the ImageNet dataset is very huge and bulky for statistical evaluations. Therefore, a few benchmark datasets will be used.

It is obvious that, for excluding contrasts, the minimal number of such datasets is three. CIFAR-10 [9], [10], NORB [11], EEACL26 datasets [12], [13] highly fit for benchmark because they are diverse and heterogeneous. Figures 1–3 represent tiny subsets of these datasets, respectively.

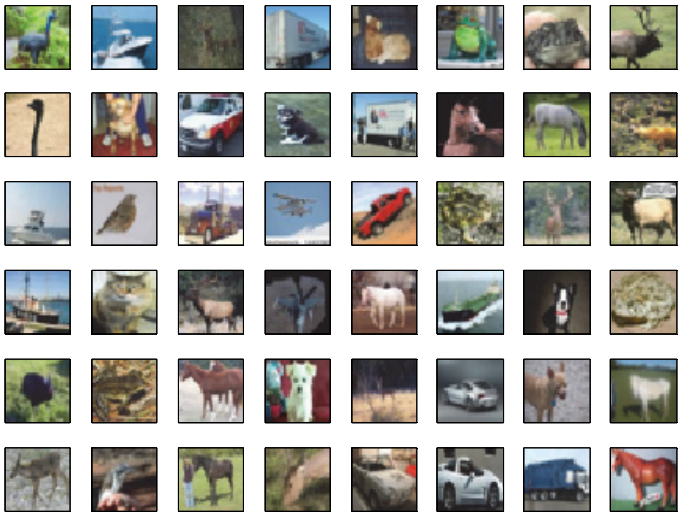

Fig. 1. A tiny subset of CIFAR-10 dataset (0.08 % of all images). Originally, this dataset contains 32 × 32 × 3 images of 10 categories. CIFAR-10 is of 60,000 colour images, where each image category is represented with 6,000 entries.

Fig. 2. A tiny subset of NORB dataset (0.012 % of all images). Originally, this dataset contains 108 × 108 × 2 images of 6 categories (5 generic categories and a background category). Each entry is a stereo-image formed with 2 images.

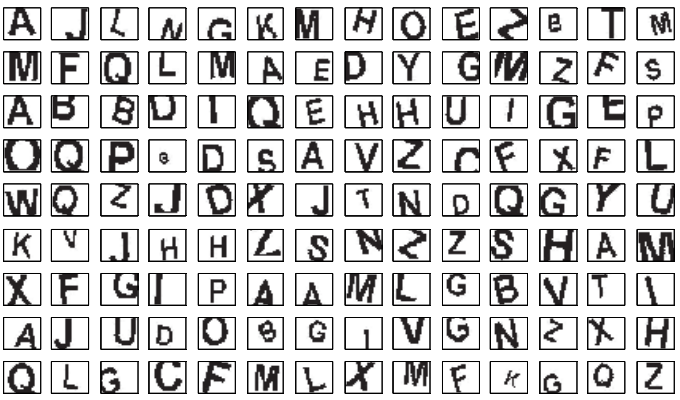

Fig. 3. A finite subset of the infinite EEACL26 dataset. This dataset is scalable, so as many entries can be generated as needed. EEACL26 entries are produced of monochrome 60 × 80 images converted to grayscale and resized to a required aspect ratio. There are 26 categories (number of alphabet letters).

CIFAR-10, NORB, and EEACL26 datasets have good distinctive properties allowing one to use them for benchmarking:

1. Their numbers of image categories are 10, 6, and 26.
2. Their numbers of colour channels are 3, 2 (stereo-images of NORB), and 1.
3. Origination of image content (NORB entries are purely artificial scenes, and EEACL26 entries are painted by a computer).
4. Types of objects to be recognised (EEACL26 entries are of the same type, but types of both CIFAR-10 and NORB entries differ much).
5. Initial image size (while CIFAR-10 entries are the tiniest, NORB and EEACL26 entries are downsampled from much bigger images).
6. Number of entries in EEACL26 is infinite, unlike in CIFAR-10 and NORB (without data augmentation [9], [14]).
7. Background clutter of EEACL26 entries is trifling, whereas background of CIFAR-10 entries varies from a monotone to a heavy clutter. NORB entries have less heavy clutter in color transitions, but there are a lot of objects that hinder in correct recognition.
8. While CNNs are trained on CIFAR-10 and NORB arduously, they are easily trained on EEACL26.

V. Image Size and Number of ConvLs in Series of All Admissible Versions for MPL Allocation

A version of MPL allocation is represented as a binary code, wherein the number of digits is equal to a number of ConvLs N_{ConvL}. Binary 1 implies that a ConvL is followed by an MPL, and binary 0 implies that a ConvL is not followed by an MPL (just as it was done in [13]).

Allocation of MPLs and their number N_{MPL} strongly depend on the image size and the number of ConvLs. As images are preferred to be square, their size is a single integer. The image size cannot be varied in a wide range, and it is preferred to be divisible, at least, by 4 or 8. Thus, let the image size be equal to 32, 48, 64, 96, 108. The last (and biggest) size is for NORB only, for saving the original entries non-resized.

Obviously, a minimal number of ConvLs is 2. Maximal one can be set to 12. Size of ConvL filters is selected to fit MPLs and subsequent layers. Numbers of ConvL filters do not confine MPLs and vice versa.

Description of CNN architecture is incomplete without integers of stride and zero-padding for ConvLs. We set them classically: the stride is 1 and zero-padding is 0.

Now, 45 versions of CNN architecture to be tested on the datasets are given in Table I. There are also 45 admissible versions for MPL allocation, although some pairs $\left(N_{\text{ConvL}}, N_{\text{MPL}}\right)$ occur more than just once. Each version has its own sizes of ConvL filters. Minimal number of MPLs is 2, which is at either $N_{\text{ConvL}} = 2$ or $N_{\text{ConvL}} = 4$ (architectures coded as “11” and “1100”). Maximal number of MPLs is 5, which occur at 7 versions of CNN architecture. However, increasing N_{ConvL} does not imply increasing N_{MPL}: there are only 3 or 4 MPLs for $N_{\text{ConvL}} \in \{9, 10, 11, 12\}$.

TABLE I

A SERIES OF ALL ADMISSIBLE VERSIONS FOR MPL ALLOCATION WITHIN 45 CNN ARCHITECTURES TO BE TESTED ON THE DATASETS BY THE IMAGE SIZE

#	CNN architecture as a binary code for ConvLs $(N_{\text{ConvL}}, N_{\text{MPL}})$	Size of ConvL filters (in order of numbering ConvLs from the CNN input)	Image size	Datasets
1	11 (2, 2)	11, 10	32	CIFAR-10, NORB, EEACL26
2		17, 15	48	
3		21, 21	64	
4		33, 31	96	
5		37, 35	108	NORB
6	111 (3, 3)	5, 5, 4	32	CIFAR-10, NORB, EEACL26
7		9, 7, 6	48	
8		9, 9, 9	64	
9		15, 14, 13	96	
10		17, 15, 15	108	NORB
11	1100 (4, 2)	5, 5, 3, 3	32	CIFAR-10, NORB, EEACL26
12		7, 6, 5, 4	48	
13	1110 (4, 3)	9, 7, 4, 4	64	
14		9, 9, 7, 6	96	
15	1111 (4, 4)	9, 9, 8, 6	108	NORB
16	11010 (5, 3)	5, 3, 2, 2, 2	32	CIFAR-10, NORB, EEACL26
17		5, 3, 3, 3, 3	48	
18	11101 (5, 4)	5, 3, 3, 3, 3	64	
19	11111 (5, 5)	5, 5, 4, 4, 2	96	
20		5, 5, 5, 3, 3	108	NORB
21	110010 (6, 3)	3, 2, 2, 2, 2, 2	32	CIFAR-10, NORB, EEACL26
22	111000 (6, 3)	5, 3, 3, 2, 2, 2	48	
23	111010 (6, 4)	5, 5, 2, 2, 2, 2	64	
24	111101 (6, 5)	5, 5, 4, 2, 2, 2	96	
25	111110 (6, 5)	5, 5, 3, 2, 2, 2	108	NORB
26	1100100 (7, 3)	3, 2, 2, 2, 2, 2, 1	32	CIFAR-10, NORB, EEACL26
27	1101000 (7, 3)	5, 3, 2, 2, 2, 2, 2	48	
28	1110010 (7, 4)	5, 5, 4, 2, 2, 2, 1	64	
29	1111010 (7, 5)	5, 5, 4, 2, 2, 2, 1	96	
30		7, 6, 4, 3, 2, 2, 1	108	NORB
31	11000010 (8, 3)	3, 2, 2, 2, 2, 2, 2, 1	32	CIFAR-10, NORB, EEACL26
32	11000100 (8, 3)	5, 3, 2, 2, 2, 2, 2, 2	48	
33	11010010 (8, 4)	5, 5, 3, 2, 2, 2, 2, 1	64	
34	11101000 (8, 4)	5, 5, 2, 2, 2, 2, 2, 2	96	
35	11110010 (8, 5)	5, 5, 3, 2, 2, 2, 2, 1	108	NORB
36	110000010 (9, 3)	5, 3, 2, 2, 2, 2, 2, 2, 2	48	CIFAR-10, NORB, EEACL26
37	110100000 (9, 3)	5, 3, 2, 2, 2, 2, 2, 2, 2	64	
38	110101000 (9, 4)	5, 3, 2, 2, 2, 2, 2, 2, 2	96	
39	111010000 (9, 4)	5, 3, 2, 2, 2, 2, 2, 2, 2	108	NORB
40	1010100000 (10, 3)	3, 3, 2, 2, 2, 2, 2, 2, 2, 2	64	CIFAR-10, NORB, EEACL26
41	1101010000 (10, 4)	5, 3, 2, 2, 2, 2, 2, 2, 2, 1	96	
42	1110100000 (10, 4)	5, 3, 2, 2, 2, 2, 2, 2, 2, 1	108	NORB
43	11010001000 (11, 4)	5, 3, 2, 2, 2, 2, 2, 2, 2, 2, 1	96	CIFAR-10, NORB, EEACL26
44	11100010000 (11, 4)	5, 3, 2, 2, 2, 2, 2, 2, 2, 2, 1	108	NORB
45	110010010000 (12, 4)	5, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1	108	

Numbers of ConvL filters are selected to be roughly optimal. Denote them by $\left\{q_{\text{filter}}(i)\right\}_{i=1}^{N_{\text{ConvL}}}$. For properly creating a benchmark analysis, all CNNs have $q_{\text{filter}}(1)=40$,

$$q_{\text{filter}}(i)=40+q_{\text{filter}}(i-1) \quad \text{for} \quad i=\overline{2,\, N_{\text{ConvL}}-1},$$

and $q_{\text{filter}}(N_{\text{ConvL}})$ is equal to the number of image categories. The filter's depth of the first ConvL is equal to the number of colour channels in the input image. Thus, this is 3 for CIFAR-10 and 1 for EEACL26. This is 1 for NORB as well due to splitting the stereo-images and averaging (it is not training set expansion [15]). The filter's depth of a successive ConvL is equal to the number of filters of the antecedent ConvL.

VI. CNN PERFORMANCE EVALUATION

CNN performance is usually treated as an error rate (a rate of inaccuracy). CNN performance is evaluated to identify the best impact of MPL allocation. Thus, the best performance is not prioritised. It is necessary to find out the fastest direction for improving the CNN performance. Therefore, let us train CNNs during 8 epochs. This number is sufficient to estimate the speed of the performance improvement [13]. For a given dataset and CNN architecture *#a* from Table I, let us denote the performance after the *p*-th epoch by $r_{\text{err}}^{<p>}(a, \text{“dataset”})$. Let us denote sets of indices tagging CNN architectures:

$$A=\left\{\overline{1,\, 45}\right\},\ A_{\text{NORB}}=\left\{5,10,15,20,25,30,35,39,42,44,45\right\}.$$

Then the following functions are to be evaluated:

$$r_{\text{err}}^{<p>}(a, \text{“CIFAR-10”}) \ \text{ by } \ a\in A\setminus A_{\text{NORB}}, \tag{1}$$

$$r_{\text{err}}^{<p>}(a, \text{“NORB”}) \ \text{ by } \ a\in A, \tag{2}$$

$$r_{\text{err}}^{<p>}(a, \text{“EEACL26”}) \ \text{ by } \ a\in A\setminus A_{\text{NORB}}. \tag{3}$$

It is clear that, even within a given dataset, the performances cannot be compared. Thus, let us take into consideration the speed of error rate descent (SoERD):

$$d_{\text{err}}(p; a, \text{“dataset”})=\frac{r_{\text{err}}^{<p-1>}(a, \text{“dataset”})}{r_{\text{err}}^{<p>}(a, \text{“dataset”})}, \quad p=\overline{2,\, 8}. \tag{4}$$

For comparison beyond a dataset, SoERD (4) is averaged:

$$\tilde{d}_{\text{err}}(a, \text{“dataset”})=\frac{1}{7}\sum_{p=1}^{7} d_{\text{err}}(p; a, \text{“dataset”}). \tag{5}$$

Nevertheless, it is a well-known fact that sometimes averages are very poor to be compared. Thus, the averaged SoERD (5) is taken with start-to-final-epoch error rate descent (SFEERD):

$$d_{\text{err}}^{<1/8>}(a, \text{“dataset”})=r_{\text{err}}^{<1>}(a, \text{“dataset”})\big/ r_{\text{err}}^{<8>}(a, \text{“dataset”}). \tag{6}$$

If training progresses successfully, both SoERD (4) and SFEERD (6) should be greater than 1. Then SFEERD (6) is expected to be far greater than the averaged SoERD (5).

CNN training is counted failed if one of the following symptoms turns to be true:

1. CNN is not trainable because

$$\left| r_{\text{err}}^{<p>}\left(a, \text{“dataset”}\right) - r_{\text{err}}^{<p+1>}\left(a, \text{“dataset”}\right) \right| < 10^{-5} \qquad (7)$$

by

$$r_{\text{err}}^{<p+1>}\left(a, \text{“dataset”}\right) > 1 - \frac{1}{Z\left(\text{“dataset”}\right)} - 10^{-5}, \qquad (8)$$

where $Z\left(\text{“dataset”}\right)$ is a number of image categories for a given dataset, $p = \overline{1,\ 7}$. Condition (7) itself is a valid argument to stop training for the CNN architecture #*a*. It is strengthened with inequality (8), which means that the training has stuck at the poorest possible rank of accuracy. Thus, there is no sense to train for the remaining number of epochs, if any.

2. CNN performs very poorly, i.e.,

$$\min_{p=1,\ 8} r_{\text{err}}^{<p>}\left(a, \text{“dataset”}\right) > 0.7 \qquad (9)$$

regardless of a dataset. For any dataset having up to a few tens of image categories, maximal accuracy less than 30 % is just catastrophic.

3. After the final training epoch, the performance is quite unsatisfactory, which is

$$r_{\text{err}}^{<8>}\left(a, \text{“dataset”}\right) > 0.7 \qquad (10)$$

regardless of a dataset. The motivation of this symptom is the same as in the previous item.

4. The training effect is minimal, i.e.,

$$r_{\text{err}}^{<8>}\left(a, \text{“dataset”}\right) > 0.95 r_{\text{err}}^{<1>}\left(a, \text{“dataset”}\right) \qquad (11)$$

regardless of a dataset. Difference in 5 % is a negligible effect of training. Such an effect might be acceptable for a complicated dataset having a few hundred image categories, but then it should be trained for thousands of epochs rather than expecting an impressive effect after the eighth epoch.

Symptom (7) by (8) helps in rejecting poor versions for MPL allocation at once, without waiting for the final training epoch to be completed. Symptoms (9) and (10) indicate poor performance. Performance may be good, but if (11) turns to be true then it implies that the training does not work effectively, so the current MPL allocation cannot be appropriate.

It has been revealed that 55 of 113 CNNs in Table I failed in training (Fig. 4). This seemingly is a much high percentage of fails (almost 49 %), but Table I includes all possible variations from the very beginning, where a large number of poorly configured CNNs should not be a surprise. Indeed, the CNN configuration is based on a number of ConvLs and rationally sizing ConvL filters, which should be well-corresponding to a number of MPLs and their allocation. As to MPL allocations, these are 29 of 45 architectures that failed. The best MPL allocation should fit any dataset, although there may be exclusions. Thus, the datasets do not have identical lists of the non-failed architectures. Figure 5 shows performance for such architectures, where

$$a \in \{6,\ 7,\ 8,\ 11,\ 12,\ 13,\ 14,\ 16,\ 17,\ 18,\ 19,\ 21,\ 22,\ 23,\ 24,\ 28,\ 29\}$$

for CIFAR-10,

$$a \in \{11,\ 12,\ 16,\ 17,\ 18,\ 19,\ 20,\ 21,\ 22,\ 23,\ 24,\ 25,\ 26,\ 27,\ 28,\ 29,\ 30,\ 31,\ 32,\ 33,\ 34,\ 35,\ 36,\ 37\}$$

for NORB, and

$$a \in \{6,\ 11,\ 12,\ 16,\ 17,\ 18,\ 19,\ 21,\ 22,\ 23,\ 24,\ 27,\ 28,\ 29,\ 32,\ 33,\ 34\}$$

for EEACL26. SoERD (4) is shown in Fig. 6.

The fastest training on EEACL26 is opposite to its simplicity. Training on CIFAR-10 and NORB progresses apparently slower (it is quite well seen from both Figs. 5 and 6), and their final results are not impressive. However, Figs. 7–9 demonstrate clearer performance of “good” CNN architectures dataset-wise. Their non-barred style is intentional – the saw-toothed polylines are perceived better for making an optimisation decision on them [16], [17].

VII. Versions for MPL Allocation that Maximise Performance

Despite those 16 “purely good” CNN architectures, there are only 12 architectures that simultaneously are good for all the three datasets. This is so, because architectures #20, #25, #30, and #35 are only for NORB with image size equal to 108. Therefore, these four architectures must not be independently considered. However, architectures #19 and #20 do not differ in MPL allocation, but just in sizing ConvL filters, and architecture #19 belongs to the series of those 12 ones (see the three sets above by the respective markers in Figs. 7–9). Architecture #24 belonging to the series differs from architecture #25 with just a shift in the last two ConvLs. The same concerns $a = 29$ (it is as #30) and $a = 34$ (the difference is severer), although the last one fails for CIFAR-10.

Except for the original NORB image size equal to 108, every image size occurs for three times among those 12 architectures. Not all are close to maximise the performance. Moreover, Figs. 7–9 visualise a lot of contradictory results by the averaged SoERD (5) and SFEERD (6). For instance, architecture #21 is at a fall for CIFAR-10, but it is at an ascent for NORB, and, surprisingly, it is at a peak for EEACL26. Similar results are for architectures #22 (a fall for NORB) and #24 (a fall for EEACL26). Therefore, none of these versions for MPL allocation is appropriate. The above-mentioned $a = 19$ takes its fall for EEACL26.

Thus, it remains to consider those tags, at which a is 11, 12, 16, 17, 18, 19, 23, 28, 29. Briefly, #11 and #12 are weaker than others to the right (but their final accuracy is comparable to others as the training start was very successful — see this in Fig. 5). Despite #16 does not give high peaks, it is reliable. Differing from #16 in sizing ConvL filters, #17 is good for just CIFAR-10. Architectures #18 and #19 would not fit simpler datasets as they are at the falls for EEACL26.

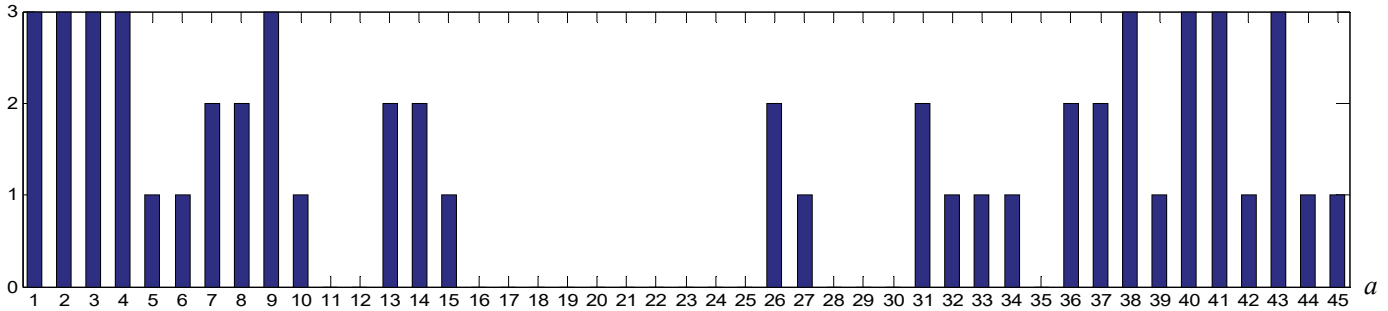

Fig. 4. A number of times when the given CNN architecture (see Table I) failed to be trained. Maximal number of fails is equal to 3, meaning that it failed for every dataset. Thus, the first four architectures, which are primitive, are rejected outright. More complicated architectures (from #36 on abscissa axis to the right) lose as well. Obviously, there are 16 "purely good" CNN architectures, which did not fail at all. A "compact" series of them from 16 to 25 is rationally attractive.

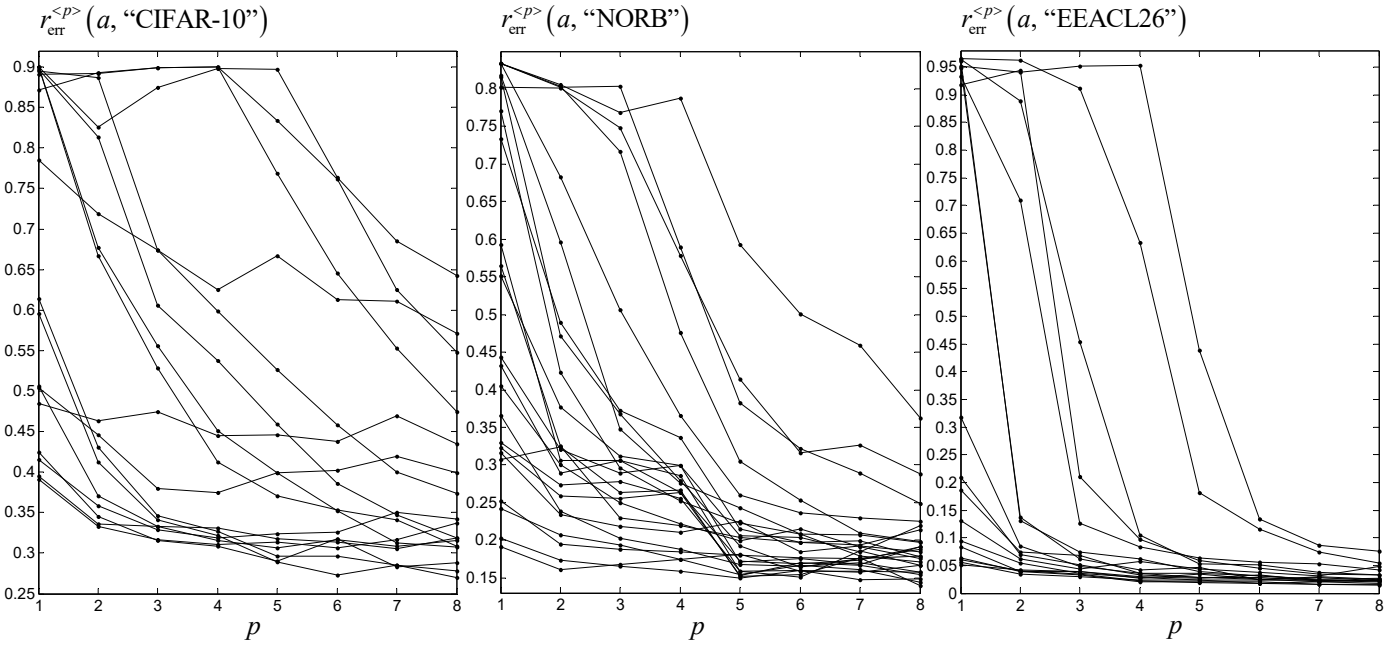

Fig. 5. Performance at the non-failed architectures for the datasets. Training on EEACL26 progresses in the fastest manner, although this dataset is the simplest one. CIFAR-10 and NORB are expectedly hard to train on them. There are three architectures, whose performance on CIFAR-10 is worse below even 50 %.

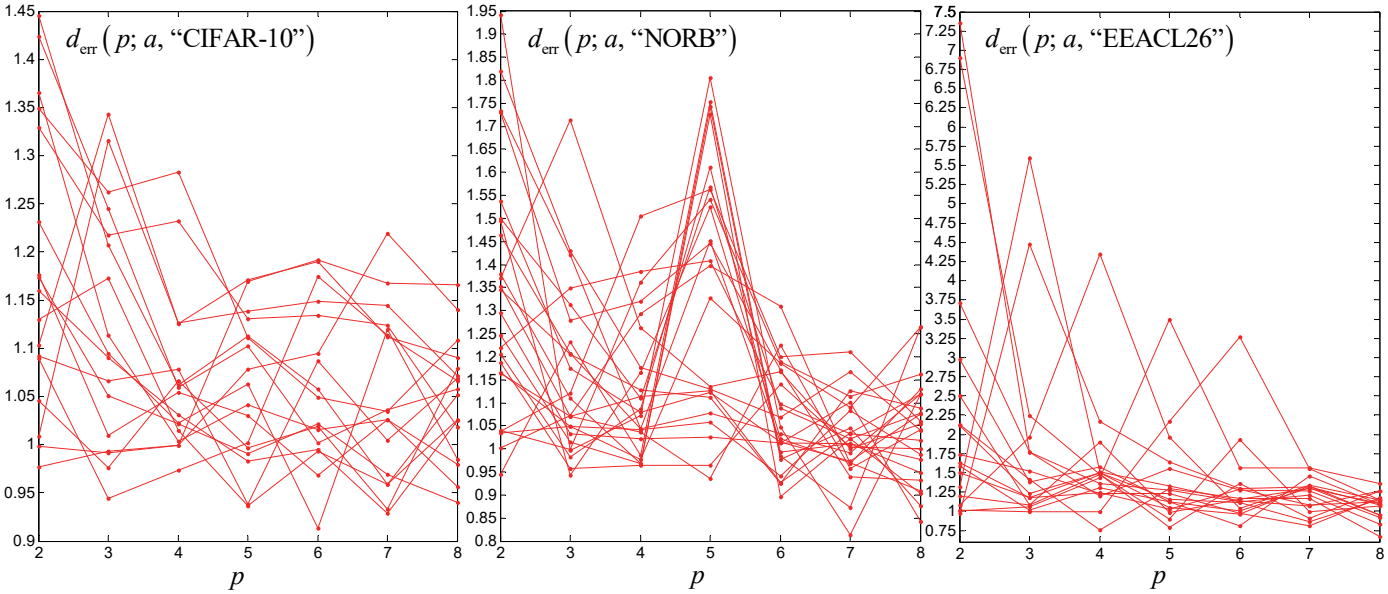

Fig. 6. SoERD (4) at the non-failed architectures. Speed for EEACL26 is far higher. However, at the final epoch, regardless of the dataset, SoERD is less than 1 for a few architectures. SoERD for CIFAR-10 does not exceed 1.5, and it does not exceed 2 for NORB. It should be noted that the strange peak after the fifth epoch at a majority of architectures for NORB is hardly explainable. On average, SoERD for all the datasets is predictably decreasing (even with that strange peak).

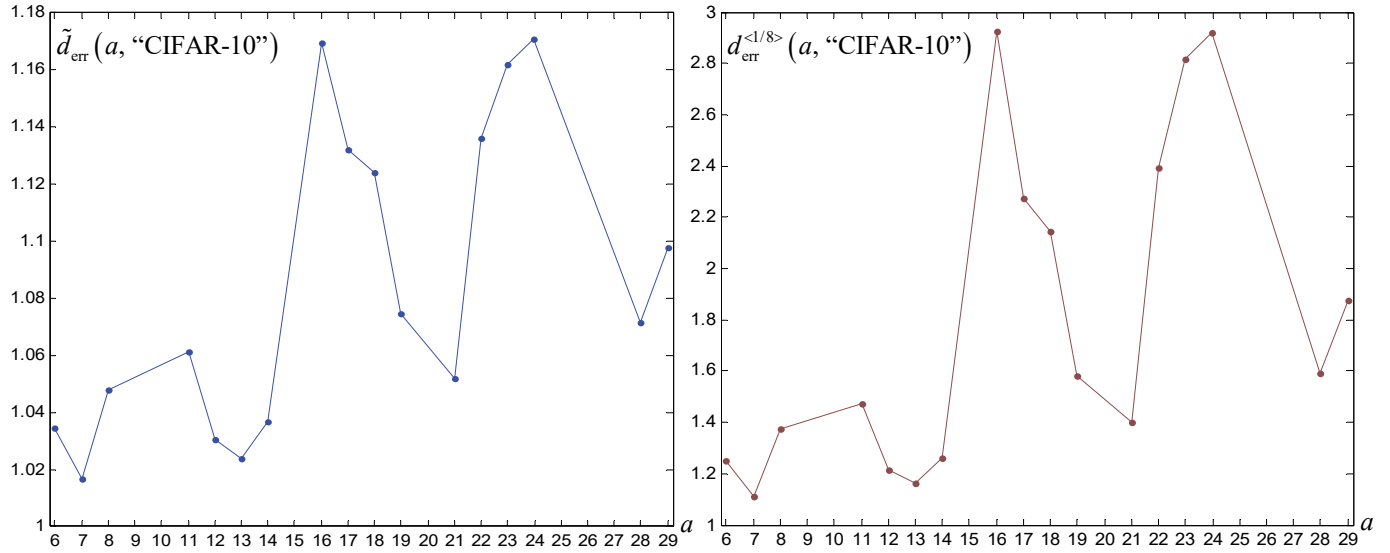

Fig. 7. The averaged SoERD (5) and SFEERD (6) for CIFAR-10. Two peaks at $a = 16$ and $a = 24$ are prominently seen. The peak at $a = 11$ is apparently weaker.

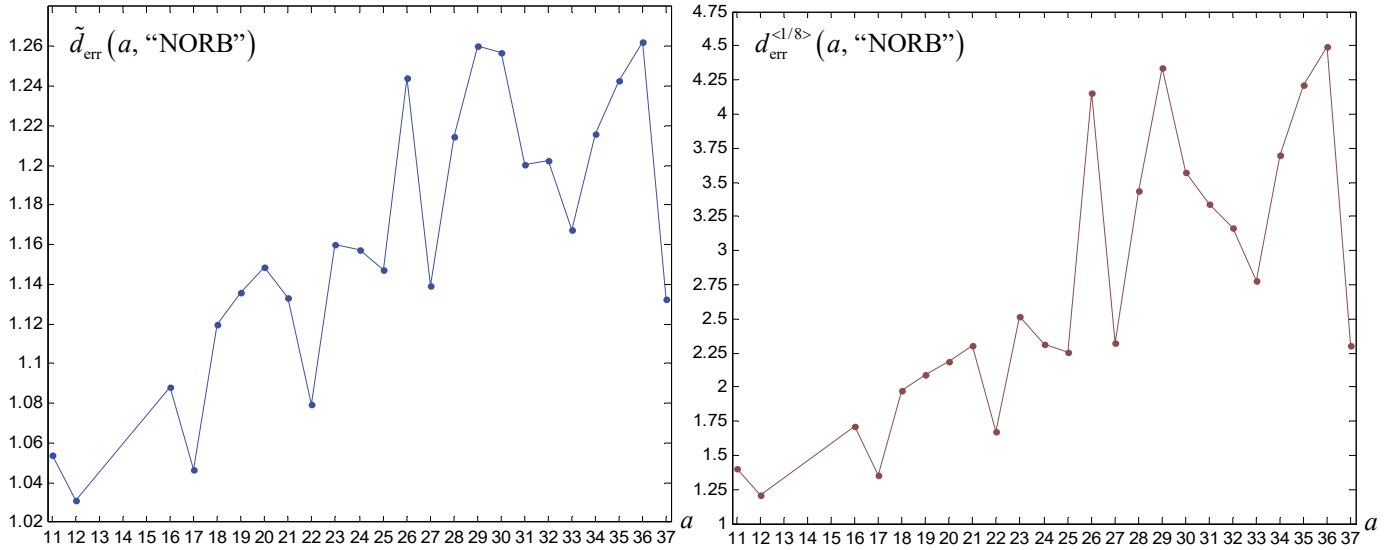

Fig. 8. The averaged SoERD (5) and SFEERD (6) for NORB. Both polylines averagely increase. Peaks at $a \in \{16, 18, 19, 20, 21, 23, 24, 26, 29, 36\}$ are fit-worthy.

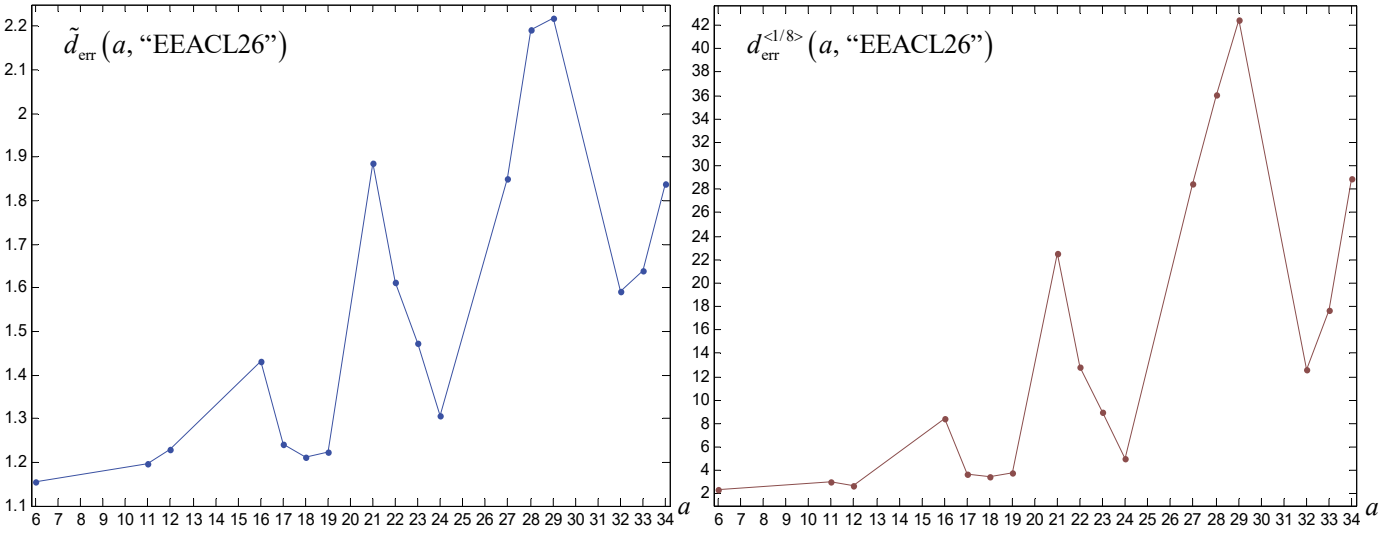

Fig. 9. The averaged SoERD (5) and SFEERD (6) for EEACL26. Four peaks at $a \in \{16, 21, 28, 29\}$ are well emphasised. Architecture #29 has the strongest impact, especially with SFEERD (6). It is important that all the mentioned ones have a similar structure – a successive group of MPLs in the starting ConvLs and the final MPL after the last but one ConvL. Moreover, most of the peaks in Figs. 7 and 8 are reached at architectures with similarly allocated MPLs.

Architectures #23, #28, and #29 are the most attractive as they are the closest to fit all three datasets. Nevertheless, it is impossible to select a single one among them that would satisfy. Thus, architecture #29 is selected with a concession for not being appropriate for CIFAR-10 (at the same image size equal to 96, #24 is much better). It is noteworthy to recall that #16 resembles #29.

VIII. Rule for the Best MPL Allocation

Despite Figs. 7–9 look like they have a common increasing saw-toothed trend, extrema (peaks and falls) of those polylines do not coincide. Hence, a rule for the best MPL allocation will be a compromise. Having analysed and compared the MPL allocations maximising the CNN performance (#16, #29, #28, #23, along with #30 and #35), it is evident that they all are of the same structure coded as "11...100...010". Consequently, the rule is to insert a few MPLs after the starting ConvLs and to insert one MPL after the last but one ConvL. Inserting an MPL after the last ConvL is the worst practice, and this is possible for CNNs with a few ConvLs (about up to 6, as in Table I) by larger images. If images are smaller and simpler (like in the original MNIST dataset and the 32×32 resized EEACL26 dataset), then CNN would probably be of 3 to 6 ConvLs (setting more, 7 or 8 ConvLs is destructive – accuracy decreases). In this case, there is no MPL inserted after the last but one ConvL [13].

IX. Discussion: Application and Limitation

Application of the rule gives an obvious benefit: appropriateness of standard 2×2 MPL allocation will considerably accelerate both training and recognition of images, not to mention better accuracy [2], [18], [19]. Naturally, the application depends upon a number of ConvLs. N_{ConvL} is determined by the image size and its complexity. Complexity of a dataset (number of image categories, sufficiency of training entries, completeness, etc.) influences N_{ConvL} as well.

The stated rule does not guarantee eventual appropriateness of MPL allocation for any dataset. Anyway, each dataset has its specifics, so versions of MPL allocation that are close to a version by the rule should be tried. Another limitation is technically computational: MPL allocation depends on how ConvL filters are sized. If the sizes of ConvL filters are adjusted after MPLs are allocated, those sizes may become non-rational. Eventually, the MPL appropriateness vanishes.

X. Conclusion

In CNNs, an appropriate number of standard 2×2 MPLs is connected to their allocation, which is recommended to be done in the form "11...100...010", where the number of MPLs before the first "zero" is about a half of N_{ConvL}. For datasets whose entries are tiny and simple images on a monotonous background, the best MPL allocation is "11...100...0" (e.g., see [15]). Versions of MPL allocations with fewer "ones" should be nonetheless tried in the first turn. The start with such a soft pooling is least destructive in relation to image specificities. Then further increment of MPLs is almost always possible, making training faster and accuracy higher (because the trained CNN receives its generalisation property [2], [3], [7], [15], [20], [21]). The represented rule is believed to be reliable owing to diversity and heterogeneousness of the used datasets. Such a small-group generalisation must ensure prevention of "overfitting" to CIFAR-10, or NORB, or EEACL26.

Acknowledgement

The research has been technically supported by the Center of Parallel Computations at Khmelnitskiy National University, Ukraine, and by the Faculty of Navigation and Naval Weapons at the Polish Naval Academy, Gdynia, Poland.

References

[1] S. Lai, L. Jin, and W. Yang, "Toward high-performance online HCCR: A CNN approach with DropDistortion, path signature and spatial stochastic max-pooling," *Pattern Recognition Letters*, vol. 89, pp. 60–66, Apr. 2017. https://doi.org/10.1016/j.patrec.2017.02.011

[2] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, "Learning Pooling for Convolutional Neural Network," *Neurocomputing*, vol. 224, pp. 96–104, Feb. 2017. https://doi.org/10.1016/j.neucom.2016.10.049

[3] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[4] H. Wu and X. Gu, "Max-Pooling Dropout for Regularization of Convolutional Neural Networks," *Lecture Notes in Computer Science*, pp. 46–54, 2015. https://doi.org/10.1007/978-3-319-26532-2_6

[5] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale Orderless Pooling of Deep Convolutional Activation Features," *Lecture Notes in Computer Science*, pp. 392–407, 2014. https://doi.org/10.1007/978-3-319-10584-0_26

[6] D. Scherer, A. Müller, and S. Behnke, "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition," *Lecture Notes in Computer Science*, pp. 92–101, 2010. https://doi.org/10.1007/978-3-642-15825-4_10

[7] B. Graham, *Fractional Max-Pooling*, May 2015.

[8] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, *Striving for simplicity: the all convolutional net*, 2015.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, iss. 6, pp. 84–90, Jun. 2017. https://doi.org/10.1145/3065386

[10] P. Date, J. A. Hendler, and C. D. Carothers, "Design Index for Deep Neural Networks," *Procedia Computer Science*, vol. 88, pp. 131–138, 2016. https://doi.org/10.1016/j.procs.2016.07.416

[11] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, vol. 2, pp. 1237–1242, 2011. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-210

[12] V. V. Romanuke, "Two-layer perceptron for classifying flat scaled-turned-shifted objects by additional feature distortions in training," *Journal of Uncertain Systems*, vol. 9, no. 4, pp. 286–305, 2015.

[13] V. V. Romanuke, "Appropriate Number and Allocation of ReLUs in Convolutional Neural Networks," *Research Bulletin of the National Technical University of Ukraine "Kyiv Polytechnic Institute"*, no. 1, pp. 69–78, Mar. 2017. https://doi.org/10.20535/1810-0546.2017.1.88156

[14] J.-J. Lv, X.-H. Shao, J.-S. Huang, X.-D. Zhou, and X. Zhou, "Data augmentation for face recognition," *Neurocomputing*, vol. 230, pp. 184–196, Mar. 2017. https://doi.org/10.1016/j.neucom.2016.12.025

[15] V. V. Romanuke, "Optimal Training Parameters and Hidden Layer Neuron Number of Two-Layer Perceptron for Generalised Scaled Object Classification Problem," *Information Technology and Management Science*, vol. 18, no. 1, pp. 42–48, Jan. 2015. https://doi.org/10.1515/itms-2015-0007

[16] V. V. Romanuke, "Training data expansion and boosting of convolutional neural networks for reducing the MNIST dataset error rate," *Research Bulletin of the National Technical University of Ukraine "Kyiv Polytechnic Institute"*, no. 6, pp. 29–34, Dec. 2016. https://doi.org/10.20535/1810-0546.2016.6.84115

[17] V. V. Romanuke, "A framework for classifier single training parameter optimization on training two-layer perceptron in a problem of turned 60-by-80-images classification," *Radio Electronics, Computer Science, Control*, no. 2, pp. 85–93, Oct. 2014. https://doi.org/10.15588/1607-3274-2014-2-13

[18] Y. Zhang and B. Shi, "Improving pooling method for regularization of convolutional networks based on the failure probability density," *Optik – International Journal for Light and Electron Optics*, vol. 145, pp. 258–265, Sep. 2017. https://doi.org/10.1016/j.ijleo.2017.07.045

[19] F. Shen, Y. Yang, X. Zhou, X. Liu, and J. Shao, "Face identification with second-order pooling in single-layer networks," *Neurocomputing*, vol. 187, pp. 11–18, Apr. 2016. https://doi.org/10.1016/j.neucom.2015.07.133

[20] J. Li, D. Zhang, J. Zhang, J. Zhang, T. Li, Y. Xia, Q. Yan, and L. Xun, "Facial Expression Recognition with Faster R-CNN," *Procedia Computer Science*, vol. 107, pp. 135–140, 2017. https://doi.org/10.1016/j.procs.2017.03.069

[21] S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin, "Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization," *Advances in Computational Mathematics*, vol. 25, no. 1–3, pp. 161–193, Jul. 2006. https://doi.org/10.1007/s10444-004-7634-z

Vadim V. Romanuke graduated from the Technological University of Podillya (Ukraine) in 2001. In 2006, he received the degree of Candidate of Technical Sciences in Mathematical Modeling and Computational Methods. The degree of Doctor of Technical Sciences in Mathematical Modeling 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 published 302 scientific articles, one monograph, one tutorial, three methodical guidelines in functional analysis, mathematical and computer modeling, conflict-controlled systems. At present, Vadim Romanuke is the scientific supervisor of a Ukrainian budget grant work concerning minimisation of water heat transfer and consumption. He is also the Head of the Department of Fitting Statistical Approximators at the Center of Parallel Computations managed by Khmelnitskiy National University.

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

E-mail: romanukevadimv@gmail.com