sciendo RIGA TECHNICAL UNIVERSITY

Applied Computer Systems

ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
December 2018, vol. 23, no. 2, pp. 141–149
doi: 10.2478/acss-2018-0018
https://content.sciendo.com

An Efficient Technique for Size Reduction of Convolutional Neural Networks after Transfer Learning for Scene Recognition Tasks

Vadim Romanuke*
Polish Naval Academy, Gdynia, Poland

***Abstract* – A complex classification task as scene recognition is considered in the present research. Scene recognition tasks are successfully solved by the paradigm of transfer learning from pretrained convolutional neural networks, but a problem is that the eventual size of the network is huge despite a common scene recognition task has up to a few tens of scene categories. Thus, the goal is to ascertain possibility of a size reduction. The modelling recognition task is a small dataset of 4485 grayscale images broken into 15 image categories. The pretrained network is AlexNet dealing with much simpler image categories whose number is 1000, though. This network has two fully connected layers, which can be potentially reduced or deleted. A regular transfer learning network occupies about 202.6 MB performing at up to 92 % accuracy rate for the scene recognition. It is revealed that deleting the layers is not reasonable. The network size is reduced by setting a fewer number of filters in the 17th and 20th layers of the AlexNet-based networks using a dichotomy principle or similar. The best truncated network with 384 and 192 filters in those layers performs at 93.3 % accuracy rate, and its size is 21.63 MB.**

***Keywords* – AlexNet, convolutional neural network, pretrained network, scene recognition, size reduction, transfer learning, truncated network.**

I. Introduction

Transfer learning has been widely used in deep learning applications since 2017 [1], [2]. A pretrained network is taken and used as a start-off point to learn a new task. If the new task is not very vast, fine-tuning a network with transfer learning is expected to be relatively faster and easier than training a network with randomly initialized weights from scratch [3], [4]. Learned features are transferred to a new task using a smaller number of training images [2], [5], [6].

In fact, pretrained networks exploited for transfer learning are complicated and consume many resources. New networks after transfer learning should be simpler and occupy less memory. However, this requires additional knowledge of how to reduce the network size without losing information.

Transfer learning is nonetheless purposed for solving tasks that have lots of features [3], [5], [6]. The tasks are not really simple themselves. One of such tasks is scene recognition [7], [8]. Scene recognition is a way harder classification task than tasks of object recognition [9], [10]. The matter is that human performance in classifying objects (like faces, cars, animals, devices, gestures, road signs, etc.) is naturally higher than in classifying indoor and outdoor places and views [8], [11], [12]. Pictures of scenes are more informative, wherein the background and its details are important. The background in pictures with objects is usually ignored.

II. Foundation and Motivation

Scenes are successfully classified by convolutional neural networks (CNNs) that allow approaching the human performance [7], [13], [14]. The best example is made with a pretrained CNN known as AlexNet [15]. A screenshot of its architecture in MATLAB is shown in Fig. 1. The CNN has five convolutional layers and three fully connected layers. AlexNet has been trained on over a million images and can classify images into 1000 object categories. The CNN has learned rich feature representations for a wide range of images (see Fig. 2). Owing to this, the AlexNet CNN is widely used for transfer learning to perform classification on a new collection of images (see Fig. 3).

A pretrained AlexNet CNN occupies about 217 MB. A fine-tuned CNN after transfer learning according to Fig. 3 occupies less memory but this decrement is not much. This is explained with the following reason. The CNN layers are transferred to a new classification task by replacing the last three layers with a fully connected layer, a softmax layer, and a classification output layer. Such three layers in AlexNet are deleted, and the replacing layers are optioned to the new task. Factually, only the new fully connected layer is re-optioned with a number of object categories of the new task. Those three last layers of the AlexNet CNN occupy just about 14.5 MB, so a new CNN size is reduced by no more than 7 %. For instance, the task with 15 object categories requires a CNN of size 202.6 MB. The size is not decreased much for the task whose number of object categories is reduced thrice – it is 202.3 MB for 5 categories.

Therefore, the transfer learning scheme in Fig. 3 is not efficient, despite it is consistent. Its inefficiency lies in that the size of a transfer learning CNN (TLCNN) is almost the same as the size of the pretrained AlexNet CNN, although the recognition task is much simpler by considering only the number of object categories. On the other hand, truly speaking, the content of tasks intending to use TLCNNs is more complicated than that in AlexNet comparing the categories in Fig. 2, for instance, to indoor and outdoor scene recognition tasks [16], [17]. Nonetheless, a grand total of information processed by CNNs like AlexNet is much bigger than a

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

common scene recognition task having up to a few tens of scene categories but no more [8], [14], [16], [18], [19].

Thus, a question is whether it is possible to obtain a TLCNN of a substantially reduced size. The TLCNN performance should not be decreased. This is very actual for tasks of classifying scenes and other complex objects, where the number of object categories is much smaller than 1000.

```
>> AlexNet.Layers
ans =
  25x1 Layer array with layers:

     1   'data'     Image Input                  227x227x3 images with 'zerocenter' normalization
     2   'conv1'    Convolution                  96 11x11x3 convolutions with stride [4  4] and padding [0  0  0  0]
     3   'relu1'    ReLU                         ReLU
     4   'norm1'    Cross Channel Normalization  cross channel normalization with 5 channels per element
     5   'pool1'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0  0  0]
     6   'conv2'    Convolution                  256 5x5x48 convolutions with stride [1  1] and padding [2  2  2  2]
     7   'relu2'    ReLU                         ReLU
     8   'norm2'    Cross Channel Normalization  cross channel normalization with 5 channels per element
     9   'pool2'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0  0  0]
    10   'conv3'    Convolution                  384 3x3x256 convolutions with stride [1  1] and padding [1  1  1  1]
    11   'relu3'    ReLU                         ReLU
    12   'conv4'    Convolution                  384 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
    13   'relu4'    ReLU                         ReLU
    14   'conv5'    Convolution                  256 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
    15   'relu5'    ReLU                         ReLU
    16   'pool5'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0  0  0]
    17   'fc6'      Fully Connected              4096 fully connected layer
    18   'relu6'    ReLU                         ReLU
    19   'drop6'    Dropout                      50% dropout
    20   'fc7'      Fully Connected              4096 fully connected layer
    21   'relu7'    ReLU                         ReLU
    22   'drop7'    Dropout                      50% dropout
    23   'fc8'      Fully Connected              1000 fully connected layer
    24   'prob'     Softmax                      softmax
    25   'output'   Classification Output        crossentropyex with 'tench' and 999 other classes
```

Fig. 1. The AlexNet CNN architecture in MATLAB. ReLU layers are typical, pooling layers are atypical, and 2 DropOut layers are inserted to prevent overfitting.

```
{'tench'                }
{'goldfish'             }
{'great white shark'    }
{'tiger shark'          }
{'hammerhead'           }
{'electric ray'         }
{'stingray'             }
{'cock'                 }
{'hen'                  }
{'ostrich'              }
{'brambling'            }
{'goldfinch'            }
{'house finch'          }
{'junco'                }
{'indigo bunting'       }
{'robin'                }
{'bulbul'               }
{'jay'                  }
{'magpie'               }
{'chickadee'            }
{'water ouzel'          }
{'kite'                 }
{'bald eagle'           }
{'vulture'              }
{'great grey owl'       }
{'European fire salamander'}
{'common newt'          }
{'eft'                  }
{'spotted salamander'   }
{'axolotl'              }
{'bullfrog'             }
{'tree frog'            }
{'tailed frog'          }
{'loggerhead'           }
{'leatherback turtle'   }
{'mud turtle'           }
{'terrapin'             }
{'box turtle'           }
{'banded gecko'         }
{'common iguana'        }
{'American chameleon'   }
{'whiptail'             }
{'agama'                }

{'custard apple'        }
{'pomegranate'          }
{'hay'                  }
{'carbonara'            }
{'chocolate sauce'      }
{'dough'                }
{'meat loaf'            }
{'pizza'                }
{'potpie'               }
{'burrito'              }
{'red wine'             }
{'espresso'             }
{'cup'                  }
{'eggnog'               }
{'alp'                  }
{'bubble'               }
{'cliff'                }
{'coral reef'           }
{'geyser'               }
{'lakeside'             }
{'promontory'           }
{'sandbar'              }
{'seashore'             }
{'valley'               }
{'volcano'              }
{'ballplayer'           }
{'groom'                }
{'scuba diver'          }
{'rapeseed'             }
{'daisy'                }
{'yellow lady's slipper'}
{'corn'                 }
{'acorn'                }
{'hip'                  }
{'buckeye'              }
{'coral fungus'         }
{'agaric'               }
{'gyromitra'            }
{'stinkhorn'            }
{'earthstar'            }
{'hen-of-the-woods'     }
{'bolete'               }
{'ear'                  }
```

Fig. 2. The first 43 and last 43 object categories of the AlexNet CNN. Some really vast objects, like “lakeside”, “seashore”, “valley”, etc., are true scenes.

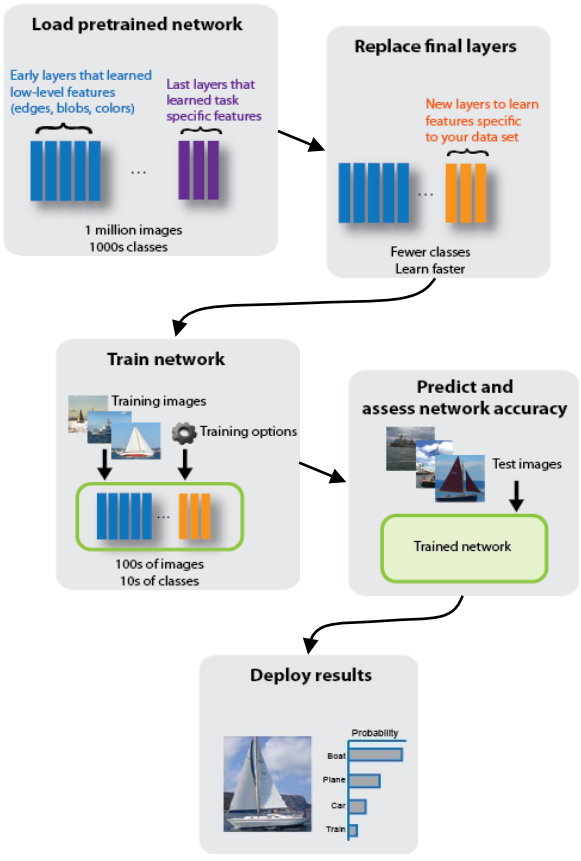

Fig. 3. A general scheme of fine-tuning a pretrained AlexNet CNN to perform classification on a new collection of images (origin: www.mathworks.com).

III. Goal and Items for Its Achievement

As the size of TLCNNs obtained by the scheme in Fig. 3 is not satisfactory, the goal is to ascertain a possibility of a TLCNN size reduction. The reduction is implied as a truncation of the pretrained CNN in the ending layers or just their simplification by decreasing numbers of filters. A modelling recognition task should be much difficult than that in AlexNet, but it should have a much smaller number of object categories for seeing an impact of the size reduction, if any. Eventually, if it is possible to simplify TLCNNs for such tasks, a method of the simplification (which factually is equivalent to the size reduction) will be stated. For achieving the said goal, the following items are to be fulfilled:

1. To substantiate a modeling recognition task.
2. To obtain TLCNNs for this task using the scheme in Fig. 3, with achieving an acceptable accuracy of recognition.
3. To try changing and truncating the ending layers in order to reduce the TLCNN size by maintaining the acceptable accuracy of recognition.
4. To state a method of the TLCNN size reduction, whichever it will be.
5. To infer from benefits, if any, of the method.

The results will be discussed paying a special attention to practical implementations. If the method works at least partially, an outlook for further research will be stated.

IV. Modelling Recognition Task

One of the known scene recognition tasks is a small dataset of 4485 grayscale images broken into 15 image categories (Fig. 4). The number of images per category is not the same (Fig. 5). It badly changes if to compare categories “bedroom”, “kitchen”, “office” to “coast”, “mountain”, “opencountry”, or “tallbuilding”. Total number of images in the training set is 4037. A validation set is 448 images. For lack of the data, the testing set is the same. Considering the variety of those 15 scenes and their complex details, the training set is very poor.

Fig. 4. A set of 100 images in the dataset. The initial size of images is various.

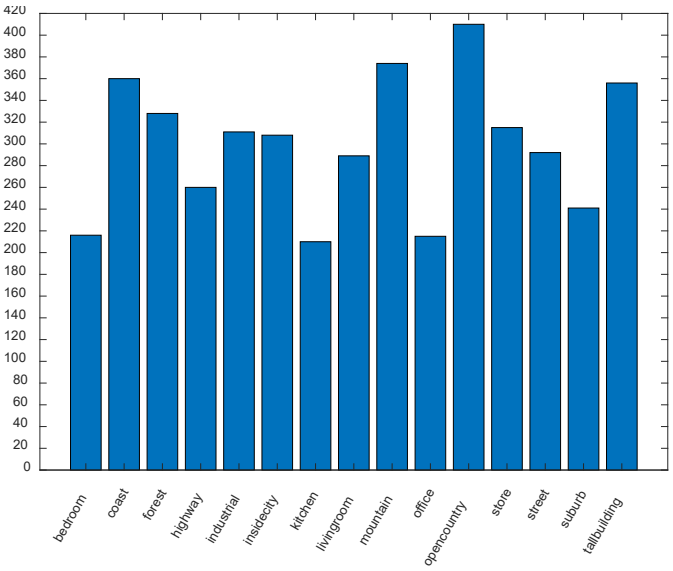

Fig. 5. The number of images per category. Its distribution appears stochastic.

To make a CNN learn from such a poor portion of the initial information, the training set must be augmented [2], [9], [20]. The augmentation is executed via horizontal and vertical shifts of the initial images. A rotation is not suitable for those images. A scaling is factually emerging from the shifts [21].

As the input of a pretrained AlexNet CNN is a colour image having three color channels, every image of the dataset must be represented as an image with three channels. Due to the lack of colour, the channels cannot differ in colour, so they may be identical. On the other hand, they may differ in shift positions like the shift augmentation would be applied to each channel separately. The whole image, after overlapping such separately shifted channels, will appear as if some noise is applied to it.

Both the training and validation sets are augmented “on the fly”, and augmented images are not saved to memory. By varying an ultimate pixel shift from 10 to 60, it appears that the best performance is achieved at a shift equal to 40. This is the shift applied to the three channels simultaneously. Shifting the channels separately does not seem to produce any effect.

V. Parameters of the Training Process

Before executing the full-scale training, its parameters should be set to close-to-efficient values. This is accomplished via launching the training process for a few epochs and selecting the values corresponding to close-to-the-highest accuracy [21], [22]. Thus, the close-to-efficient parameters of the training process are the following:

1. A solver for training CNNs is the stochastic gradient descent with momentum optimizer.
2. A size of the mini-batch is 45 (the min-batch is a subset of the training set that is used to evaluate the gradient of the loss function and update the weights).
3. Maximum number of epochs is between 28 (this is for fine-tuning a pretrained AlexNet CNN by the scheme in Fig. 3) and 128 (this is for cases when the ending layers of TLCNNs are changed and truncated).

4. An initial learning rate is 0.001 (it is not too high in order to prevent diverging).
5. A momentum is 0.9 (this is 90 % contribution of the parameter update of the previous iteration to the current one).
6. The learning rate is dropped during training: this rate is multiplied by 0.975 after every epoch.
7. A factor for $\mathbb{L}_2$ regularization (weight decay) is 0.0001.

Setting an environment for training is optional. It can be either CPU or GPU, depending on whether GPUs are available and consistent. Obviously, training on even a single GPU is expected to be faster than training on a multicore CPU.

VI. Regular TLCNNs

Training TLCNNs according to Fig. 3 (regular TLCNNs) is not very long. Owing to the pretrained CNN consisting now of 22 layers (without those ending three layers, which have been factually deleted), the training and validation accuracies expectedly increase fast and go into saturation (Fig. 6). The saturation rate is up at 92 %. It is never achieved but approached since just a few epochs. Such a good performance from the very starting epochs is explained by that the first layer filters have learned edge-like features from training data of AlexNet (see them in Fig. 7).

Fig. 6. A screenshot of the training process progress for regular TLCNNs. The training loses after the 11th epoch. The TLCNN size is approximately 202.6 MB.

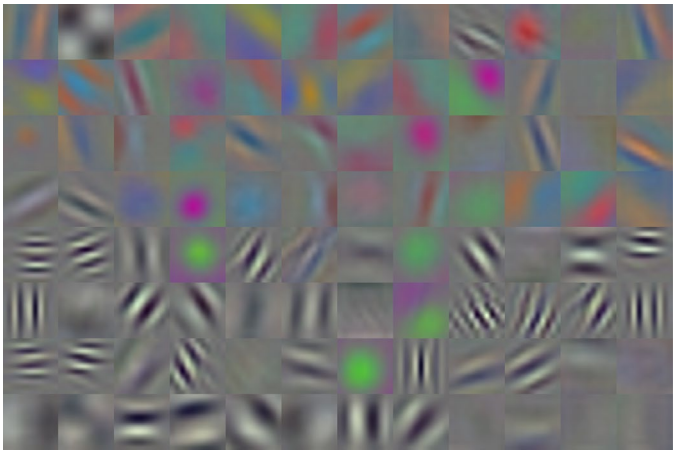

Fig. 7. The first layer weights have a well-defined structure used in TLCNNs.

Accuracy at 91.07 % rate achieved under the set above parameters of the training process is acceptable. It still may be improved by additionally adjusting the learning rate drop factor. Nevertheless, the accuracy improvement is not going to be significant. Henceforward, we are going to try changing and truncating the ending layers under the accepted configuration of the training process in order to reduce the TLCNN size and to approach the achieved accuracy.

VII. TLCNNs with Deeper Truncations

Regular TLCNNs have three fully connected layers. They have 4096, 4096, and 15 filters. Could we decrease those numbers in the 17th and 20th layers without an accuracy decrement? This question is addressed to a substantial reduction of the layer volume, not a few percent decrement of the number of the convolutional layer filters. For instance, the 17th and 20th layers may be tried with 2048 and 1024 filters.

A more radical variant is to delete one or two fully connected layers. However, when a one fully connected layer is deleted, the TLCNN has size of 143 MB, but it is trained then less efficiently, with a slightly dropped accuracy (Fig. 8). Therefore, deletion of fully connected layers is not reasonable.

Decreasing the numbers of filters in the 17th and 20th layers should be done in a way reminding a dichotomy or tetrachotomy [23], with multiplicity of 2 to some integer power. As initially those layers are of 4096 filters both, the following combinations are tried: 1024 and 512, 768 and 384, 512 and 256, 384 and 192, 256 and 128, and so on. It appears that the best combination of them is 384 and 192 filters (Fig. 9).

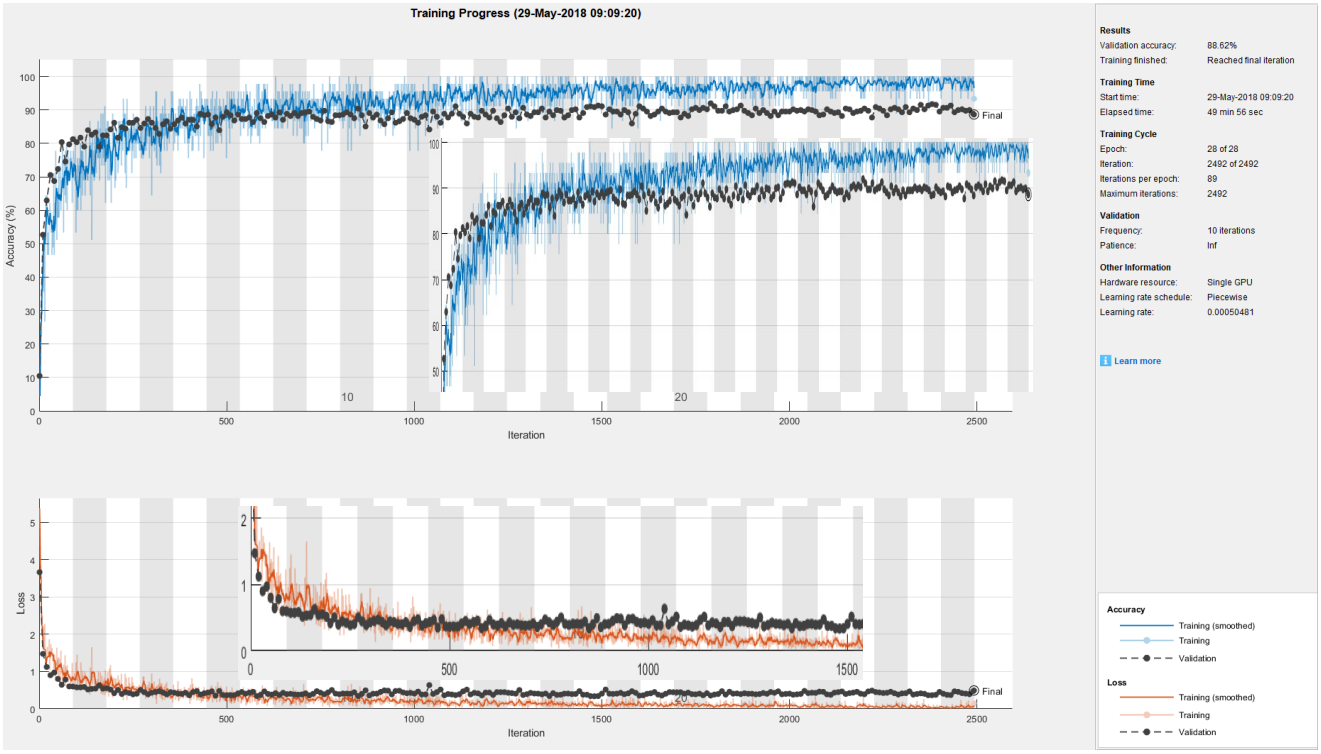

Fig. 8. The truncated TLCNN with 22 layers is trained less efficiently than a regular TLCNN with 25 layers (compare it to Fig. 6), although the odds are small.

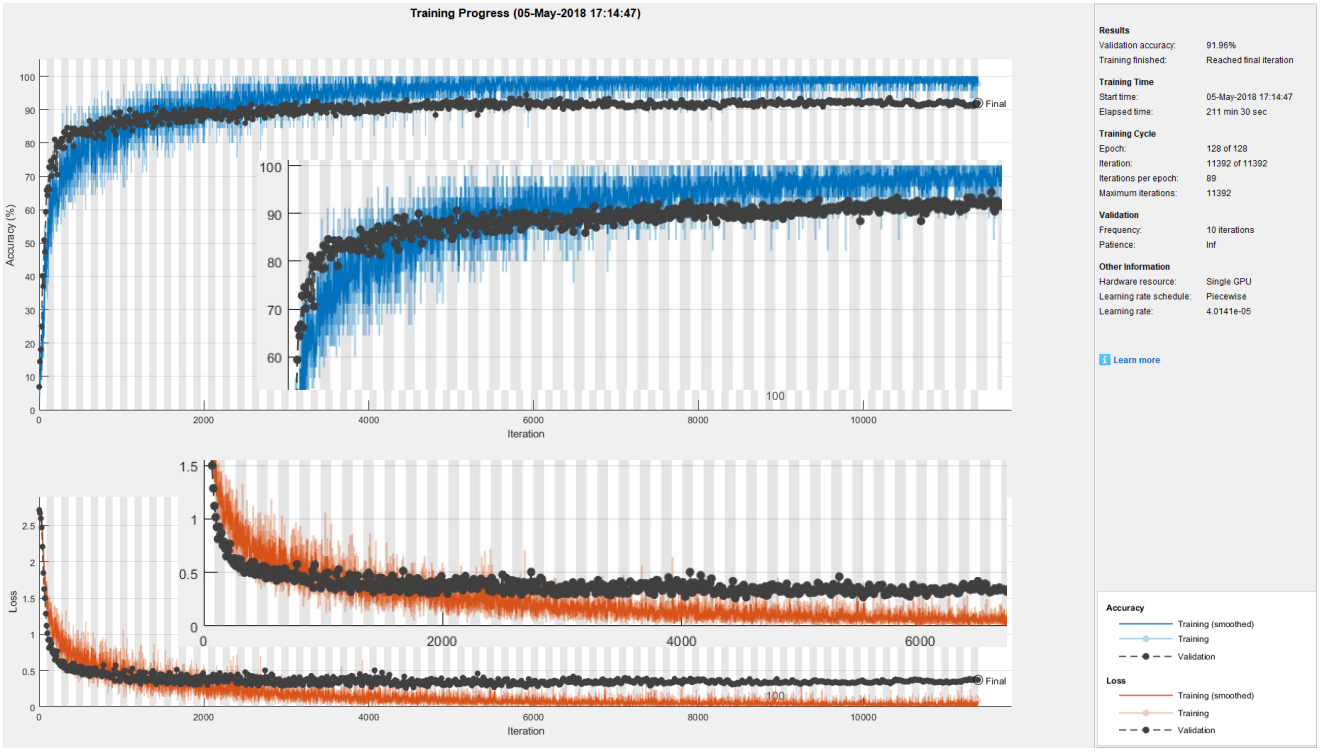

Fig. 9. The training progress for a truncated TLCNN with 384 and 192 filters in its 17th and 20th layers. High accuracies are achieved before the final accuracy.

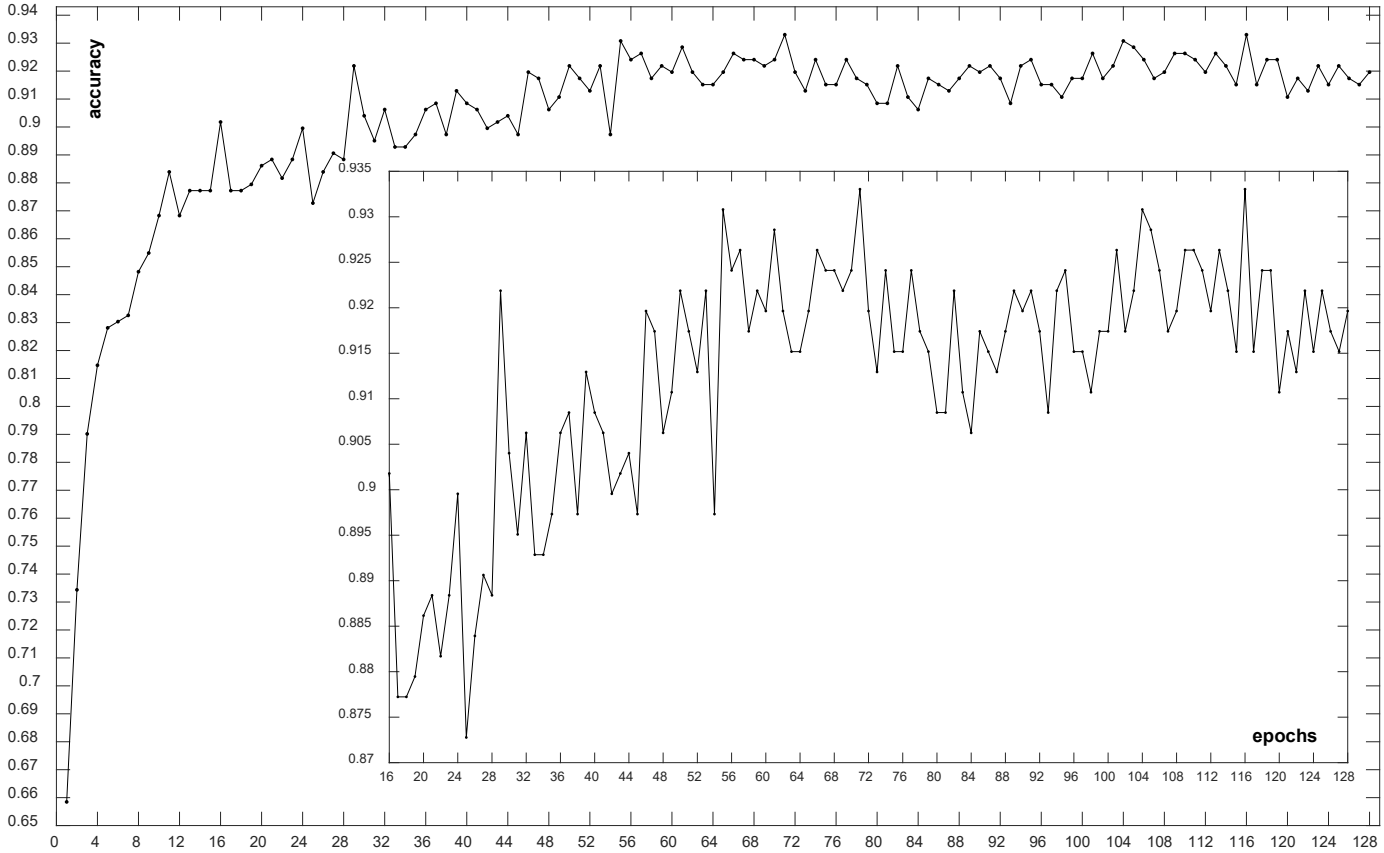

Fig. 10. The accuracy of the TLCNN with 384 and 192 filters in the 17th and 20th layers against the number of epochs. The best accuracy is first achieved after the 71st epoch, whereupon the same accuracy is achieved after the 116th epoch. Thus, the training process could have been stopped right after the 71st epoch.

The size of the TLCNN with 384 and 192 filters in the 17th and 20th layers is 21.63 MB, whereas the accuracy is even better than the accuracy of a regular TLCNN. A truncated TLCNN performing at the highest accuracy is selected amongst TLCNNs produced after those 128 epochs. The best truncated TLCNN in Fig. 9 happens two times: after the 71st and 116th epochs (Fig. 10). This is an outstanding positive impact of almost 9.4 times reducing the size of TLCNN.

The top achieved 93.3 % accuracy rate is really close to the human performance on that 15 scene dataset conceding only about 1.5 %. Figure 10 shows that the accuracy rate does not drop below 90.5 % since the 55th epoch. Any TLCNN is a good scene classifier since that epoch. Besides, owing to the reduction, truncated TLCNNs operate faster that regular ones.

VIII. Method of the TLCNN Size Reduction

Regarding AlexNet-based TLCNNs, the following method is stated for the TLCNN size reduction:

1. Only two fully connected layers can be changed by setting a fewer number of their filters.
2. The number of the filters can be decreased simultaneously by using a principle reminding a dichotomy or tetrachotomy: decrements must be equal to 2^{n-1} or 2^{n-2}, where 2^n is the current number of filters.
3. If the accuracy after such decrements drops dramatically, a gradual decreasing must be tried. Generally, the number of filters is decreased by 2^{n-m}, where $m = \overline{1, n-1}$.
4. The decrements are canceled when the accuracy drops below an acceptable rate, and the last successfully truncated TLCNN is returned/stored.

This method does not ensure increasing the accuracy. Nonetheless, it ensures that the TLCNN size, for tasks similar to the recognition of 15 scene categories, will be reduced, whichever the reduction is.

IX. Discussion

Along with reducing the TLCNN size, a kind of faster training is benefited from the stated method. However, it is not as fast as it could have been if the initial size were kept. The reason is that, after reduction, TLCNN starts re-learning some specific features previously stored in the whole layers before reducing them. Moreover, a greater number of epochs may be required for appropriately training the TLCNN in order to achieve an acceptable rate of the recognition accuracy. Thus, the training process may factually take a longer period than it would have taken for TLCNNs by the scheme in Fig. 3.

The example of recognizing 15 scene categories is a hard-learning recognition task, which is unlikely to be solved at 93 % accuracy rate without transfer learning. Similar tasks, if having more categories at roughly the same density of the training set

per category, will be solved with an apparently less impact of the TLCNN size reduction. Indeed, a greater number of categories will require smaller decrements of the number of the filters in the 17th and 20th layers. However, the studied example is a close-to-average hard-learning recognition task for transfer learning. Some tasks have a fewer scene/object/image category, where an impact of the TLCNN size reduction will be greater than 10. Such tasks, for example, are generated from the benchmark datasets initially provided for semantic image segmentation (SIS) tasks. The seeming simplicity of recognising scenes in the SIS-generated tasks dissolves in a limited number of training samples (see Fig. 11). Besides, some scenes may belong to a few categories (Fig. 12). As a result, the AlexNet CNN performs even poorer than a TLCNN (Fig. 13). Then the numbers of the filters are decreased without affecting the accuracy (Fig. 14), although the impact of the TLCNN size reduction stands here apart (Fig. 15). But the limit of such an impact does exist. This is 21.8 (9.294 MB against 202.6 MB), when two fully connected layers are deleted and the earlier 16 layers are non-removable (see the CNN architecture in Fig. 1). Nevertheless, subsequent training of so deeply truncated TLCNNs fairly fails.

Fig. 11. A validation set of 52 images in the dataset with just 4 scene categories. The dataset is of 534 images [20]. This SIS-generated task has only 70 and 48 training samples for categories "house-about" and "road", respectively. The other two categories, consisting of 235 and 181 images, have many similar entries.

Fig. 12. A validation set of 70 images in the dataset with 9 scene categories. The dataset is of 689 images [9]. The scene categories are unequally intertwined.

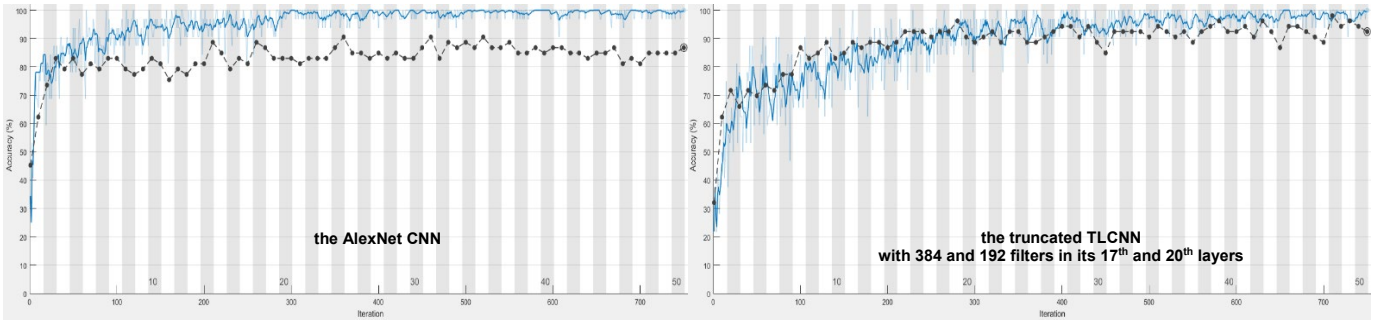

Fig. 13. The training progress of the AlexNet CNN (202.39 MB) against a TLCNN (21.57 MB) for the task by Fig. 11. The TLCNN size reduction impact is 9.4.

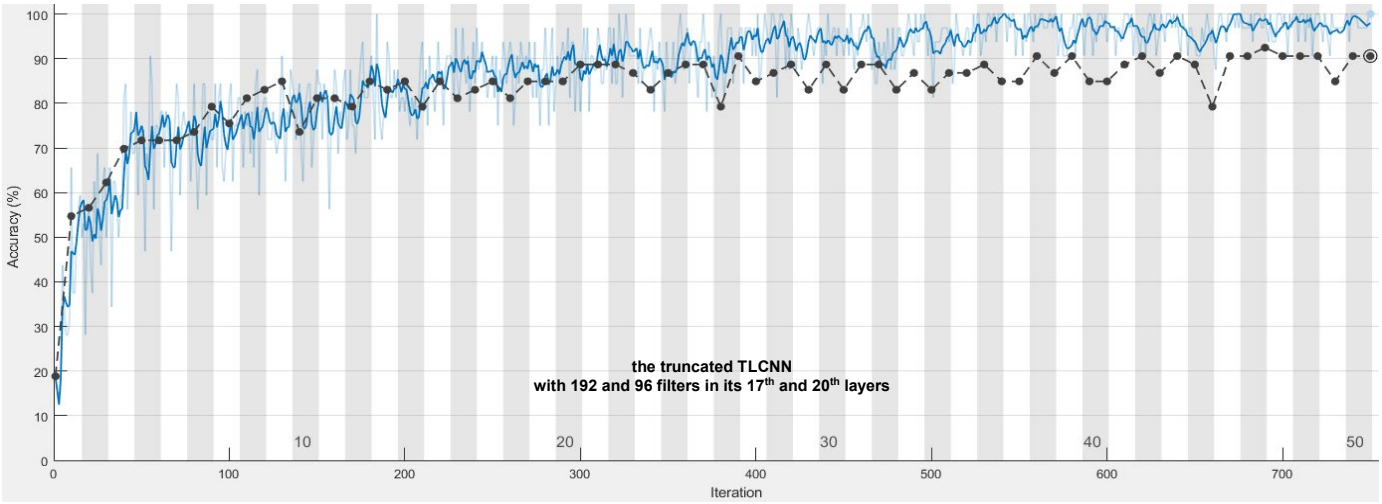

Fig. 14. The training progress of a TLCNN (15.1 MB) for the task by Fig. 11. The impact is 13.4, and the accuracy is greater than that of the AlexNet in Fig. 13.

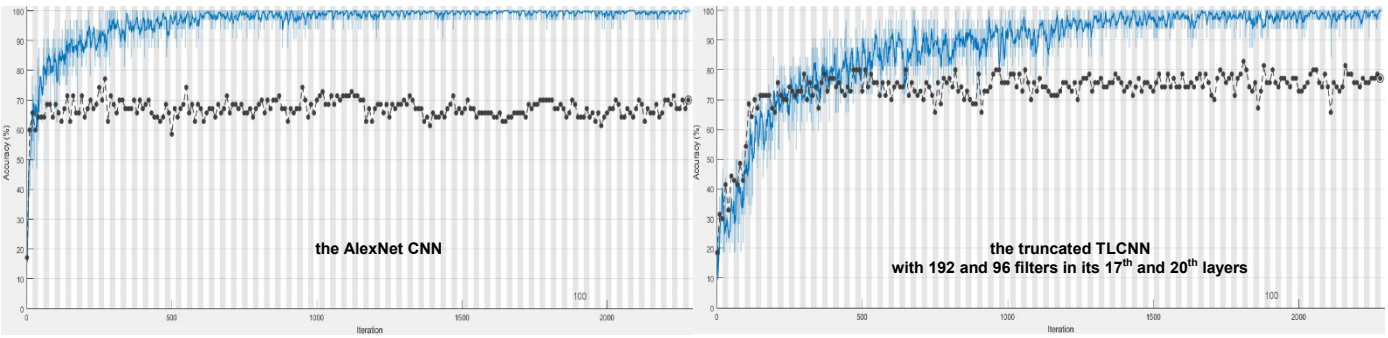

Fig. 15. The training progress of the AlexNet (202.48 MB) against a TLCNN (15.11 MB) for the task by Fig. 12. The impact is 13.4 at the improvable accuracy.

X. Conclusion

Despite the seeming simplicity of transfer learning, it cannot be efficiently executed as visualised in Fig. 3. The TLCNN size is reduced by setting a fewer number of the filters in the 17th and 20th layers of the AlexNet-based TLCNNs. In general, the number of filters in fully connected layers preceding the last fully connected layer can be decreased. The decrements are made from the end of a TLCNN starting from a half of the last but one layer or halves of layers, moving thus to early layers. If the accuracy drops a little after starting decrements, smaller portioned decrements must be made.

The stated method serves for tasks with a few tens of object categories or fewer, whose training set is pretty poor but has complex objects. A classic example is the studied scene recognition task, although tasks with images of outdoor scenes may imply up to a hundred or a few hundred categories. Such cases may either be solved with non-reducible TLCNNs or even not fit to apply transfer learning (i.e., a CNN should be trained from scratch like it should be for the task by Fig. 12).

Since setting hyperparameters of CNNs and parameters of their training has many close-to's, the method of the TLCNN size reduction is not the only feasible [24], [25]. It is plausible that changing some training parameters may allow decreasing numbers of filters further. Two of those parameters are the learning rate drop factor and $\mathbb{L}_2$ regularization factor [26], but this question needs a separate study.

REFERENCES

[1] S. M. Salaken, A. Khosravi, T. Nguyen, and S. Nahavandi, "Extreme learning machine based transfer learning algorithms: A survey," *Neurocomputing*, vol. 267, pp. 516–524, 2017. https://doi.org/10.1016/j.neucom.2017.06.037

[2] D. Han, Q. Liu, and W. Fan, "A new image classification method using CNN transfer learning and web data augmentation," *Expert Systems with Applications*, vol. 95, pp. 43–56, 2018. https://doi.org/10.1016/j.eswa.2017.11.028

[3] L. Wang, L. Ge, R. Li, and Y. Fang, "Three-stream CNNs for action recognition," *Pattern Recognition Letters*, vol. 92, pp. 33–40, 2017. https://doi.org/10.1016/j.patrec.2017.04.004

[4] V. Campos, B. Jou, and X. Giró-i-Nieto, "From pixels to sentiment: Fine-tuning CNNs for visual sentiment prediction," *Image and Vision Computing*, vol. 65, pp. 15–22, 2017. https://doi.org/10.1016/j.imavis.2017.01.011

[5] L. H. S. Vogado, R. M. S. Veras, F. H. D. Araujo, R. R. V. Silva, and K. R. T. Aires, "Leukemia diagnosis in blood slides using transfer learning in CNNs and SVM for classification," *Engineering Applications of Artificial Intelligence*, vol. 72, pp. 415–422, 2018. https://doi.org/10.1016/j.engappai.2018.04.024

[6] A. Khatami, M. Babaie, H. R. Tizhoosh, A. Khosravi, T. Nguyen, and S. Nahavandi, "A sequential search-space shrinking using CNN transfer learning and a Radon projection pool for medical image retrieval," *Expert Systems with Applications*, vol. 100, pp. 224–233, 2018. https://doi.org/10.1016/j.eswa.2018.01.056

[7] X. Cheng, J. Lu, J. Feng, B. Yuan, and J. Zhou, "Scene recognition with objectness," *Pattern Recognition*, vol. 74, pp. 474–487, 2018. https://doi.org/10.1016/j.patcog.2017.09.025

[8] X. Song, S. Jiang, L. Herranz, Y. Kong, and K. Zheng, "Category co-occurrence modeling for large scale scene recognition," *Pattern Recognition*, vol. 59, pp. 98–111, 2016. https://doi.org/10.1016/j.patcog.2016.01.019

[9] S. Gould, R. Fulton, and D. Koller, "Decomposing a scene into geometric and semantically consistent regions," *Proceedings of 2009 IEEE 12th International Conference on Computer Vision*, pp. 1–8, 2009. https://doi.org/10.1109/iccv.2009.5459211

[10] Z. Ding, M. Shao, and Y. Fu, "Incomplete multisource transfer learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 2, pp. 310–323, 2018. https://doi.org/10.1109/TNNLS.2016.2618765

[11] H. Zhao, Q. Liu, and Y. Yang, "Transfer learning with ensemble of multiple feature representations," *Proceedings of 2018 IEEE 16th International Conference on Software Engineering Research, Management and Applications (SERA)*, pp. 54–61, 2018. http://doi.ieeecomputersociety.org/10.1109/SERA.2018.8477189

[12] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability for a generic ConvNet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 9, pp. 1790–1802, 2016. https://doi.org/10.1109/TPAMI.2015.2500224

[13] S. Bai, and H. Tang, "Softly combining an ensemble of classifiers learned from a single convolutional neural network for scene categorization," *Applied Soft Computing*, vol. 67, pp. 183–196, 2018. https://doi.org/10.1016/j.asoc.2018.03.007

[14] P. Tang, H. Wang, and S. Kwong, "G-MS2F: GoogLeNet based multi-stage feature fusion of deep CNN for scene recognition," *Neurocomputing*, vol. 225, pp. 188–197, 2017. https://doi.org/10.1016/j.neucom.2016.11.023

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 2, pp. 84–90, 2017. https://doi.org/10.1145/3065386

[16] C. Wang, J. Yu, and D. Tao, "High-level attributes modeling for indoor scenes classification," *Neurocomputing*, vol. 121, pp. 337–343, 2013. https://doi.org/10.1016/j.neucom.2013.05.032

[17] S. Bai, "Growing random forest on deep convolutional neural networks for scene categorization," *Expert Systems with Applications*, vol. 71, pp. 279–287, 2017. https://doi.org/10.1016/j.eswa.2016.10.038

[18] B.-J. Han, and J.-Y. Sim, "Saliency detection for panoramic landscape images of outdoor scenes," *Journal of Visual Communication and Image Representation*, vol. 49, pp. 27–37, 2017. https://doi.org/10.1016/j.jvcir.2017.08.003

[19] J.-T. Lee, H.-U. Kim, C. Lee, and C.-S. Kim, "Photographic composition classification and dominant geometric element detection for outdoor scenes," *Journal of Visual Communication and Image Representation*, vol. 55, pp. 91–105, 2018. https://doi.org/10.1016/j.jvcir.2018.05.018

[20] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," *Proceedings of 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1253–1260, 2010. https://doi.org/10.1109/cvpr.2010.5539823

[21] 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, 2017. https://doi.org/10.20535/1810-0546.2017.1.88156

[22] 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, 2015. https://doi.org/10.1515/itms-2015-0007

[23] V. V. Romanuke, "Interval uncertainty reduction via division-by-2 dichotomization based on expert estimations for short-termed observations," *Journal of Uncertain Systems*, vol. 1, no. 12, pp. 3–21, 2018.

[24] J. Yang, S. Li, and W. Xu, "An iterative transfer learning based classification framework," *Proceedings of 2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2018. https://doi.org/10.1109/IJCNN.2018.8489471

[25] X. Liu, Z. Liu, G. Wang, Z. Cai, and H. Zhang, "Ensemble transfer learning algorithm," *IEEE Access*, vol. 6, pp. 2389–2396, 2018. https://doi.org/10.1109/ACCESS.2017.2782884

[26] Y. Liu, D. Yang, and C. Zhang, "Relaxed conditions for convergence analysis of online back-propagation algorithm with $\mathbb{L}_2$ regularizer for Sigma-Pi-Sigma neural network," *Neurocomputing*, vol. 272, pp. 163–169, 2018. https://doi.org/10.1016/j.neucom.2017.06.057

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 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, semantic image segmentation, and control engineering based on statistical correspondence. He is the author of 323 scientific articles, one monograph, one tutorial, three methodical guidelines in functional analysis, mathematical and computer modeling, Master Thesis preparation, 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. Now he directs a branch of fitting statistical approximators at the Center of Parallel Computations in Khmelnitskiy, Ukraine.

Address for correspondence: 69 Śmidowicza Street, Gdynia, Poland, 81-127.
E-mail: romanukevadimv@gmail.com
ORCID iD: https://orcid.org/0000-0003-3543-3087