Choice of Neural Network Architecture when Recognizing Objects that do not Have High-Level Features

Gregory Malyshev[1], Vyacheslav Andreev[2], Olga Andreeva[2], Oleg Chistyakov[1] and Dmitriy Sveshnikov[1]

[1] *JSC «Afrikantov OKBM», Burnakovsky passage 15, Nizhny Novgorod, 603074, Russia*
[2] *Nizhny Novgorod state technical university n.a. R.E. Alekseev, Minina 24, Nizhny Novgorod, 603950, Russia*

Abstract

This article explores the capabilities of pretrained convolutional neural networks in relation to the problem of recognizing defects for which it is impossible to identify any abstract features. The results of training the convolutional neural network AlexNet and the fully connected classifier of the VGG16 network are compared. The efficiency of using a pretrained neural network in the problem of defect recognition is demonstrated. A graph of the change in the proportion of correctly recognized images in the process of training a fully connected classifier is presented. The article attempts to explain the efficiency of a fully connected neural network classifier trained on a critically small training dataset with images of defects. The work of a convolutional neural network with a fully connected classifier is investigated. The classifier allows for classification into five categories: «crack» type defects, «chip» type defects, «hole» type defects, «multi hole» type defects and «defect-free surface». The article provides examples of convolutional network activation channels, visualized for each of the five categories. The signs of defects on which the activation of the network channels takes place are formulated. The classification errors made by the network are analyzed. The article provides predictive probabilities, below which the result of the network operation can be considered doubtful. Practical recommendations for using the trained network are given.

Keywords

convolutional neural networks, pretrained neural networks, activation channels, image recognition, defects

1. Introduction

Currently, the most advanced image recognition tool is convolutional neural networks [1-4]. The use of modern neural network architectures [5-8] and a large training set will certainly allow obtaining a high percentage of correctly recognized images. Such studies are no longer original. At the same time, identifying tasks in which it is quite possible to get by with networks of simple architecture is a hot topic.

This article examines the operation of a convolutional neural network, which allows the classification of defects on products into five categories: defects of the "crack" type, defects of the "chip" type, defects of the "single pore" type, defects of the "accumulation of pores" type and "defect-free surface".

For each of the five classified categories, the researchers only had 55 images, meaning the entire training set consisted of only 275 images. Seventy five images (15 images per category) were used to validate the network during the training (validation) phase. The set of validation already at the training stage allows us to track the epoch from which the network retraining begins [1, 9]. In addition, there

GraphiCon 2021: 31st International Conference on Computer Graphics and Vision, September 27-30, 2021, Nizhny Novgorod, Russia
EMAIL gsmalyshev@okbm.nnov.ru (G. Malyshev); vyach.andreev@mail.ru (V. Andreev); andreevaov@gmail.com (O. Andreeva); gsmalyshev@okbm.nnov.ru (O. Chistyakov); budnikov@okbm.nnov.ru (D. Sveshnikov);
ORCID: 0000-0002-8147-988X (G. Malyshev); 0000-0002-7557-352X (V. Andreev); 0000-0001-9581-3028 (O. Andreeva); 0000-0002-6515-9691 (O. Chistyakov); 0000-0002-2152-5346 (D. Sveshnikov);

were 50 test images (10 images per category) available, which were used to test the network performance after it was trained.

2. Choosing a convolutional network architecture

When choosing a network architecture, the researchers proceeded from the fact that the initial layers of the convolutional neural network highlight the most generalized (local) features (for example, boundaries and textures) in the image, while deeper layers highlight abstract concepts, that is, high-level features (such as "cat's nose" or "bird's feather") [9]. When it comes to defects, it is rather difficult to talk about any abstract signs of defects, since defects look very amorphous.

Thus, the main information about defects should be laid down in low-level features, the most complex of which can be, for example, broken lines (a sign of a crack), darkening on the surface or large accumulations of small dark spots (a sign of pore accumulation), violation of strict geometry at the edges. Products (a sign of chipping), a single spot (a sign of a separate pore), large surface areas with a uniform texture (a sign of a defect-free surface). These features are far from abstract, therefore, to identify such features, it is enough to use a convolutional network [10, 11] with a sufficiently small number of convolutional layers (no more than fifteen).

Complex, abstract features (which are not detected in a problem with defect recognition), characteristic of specific classes, are “wired up” in deep layers. Therefore, to solve the problem of recognizing defects, we can remove not only the fully connected classifier [9], but also the deep convolution layers. The parameters of the convolutional basis of the network must be frozen in the process of training a new classifier, that is, only weights and thresholds [1, 9] of a fully connected classifier will be trained. Naturally, you can also retrain the parameters of the convolutional basis or its individual blocks. But this approach leads to significant time costs.

Initially, an attempt was made to solve the problem of defect recognition using the AlexNET network [12]. The network was trained from scratch using the functions of the Keras library written in Python. To initialize the weights, a normal distribution with zero mean and a standard deviation of 0,1 was used. The initial thresholds were set to 0.1. The RMSprop method [1] was chosen as the optimization method (gradient descent method). When training the network using Keras library functions, the initial learning rate was set to $2 \cdot 10^{-5}$, and the rest of the parameters of the RMSprop algorithm were left by default (these parameters can be found in the description of the Keras library functions, for example, here [13]).

Each iteration, five images (minibatch) from the training set were fed to the network input. Already after eight epochs of training, the effect of overfitting began to manifest itself: losses on the validation set began to increase, so training was stopped. The trained network showed poor performance results: the share of correctly recognized images from the test set was only 72%. Such a low recognition accuracy can be explained (in addition to the small volume of the training set) by the too primitive architecture of the network, as well as by ineffective initial initialization of weights and thresholds [14-16]. In [1], it was shown that even the most successful combinations of the initial initialization of the neural network parameters and the gradient descent algorithm can significantly increase the network learning rate, but will not give a serious gain in the accuracy of image recognition. That is, even using the pre-trained AlexNET network does not guarantee high recognition accuracy after additional training.

Subsequently, the problem of recognizing defects was solved using the VGG16 network [17-18]. To avoid the difficulties associated with the initial initialization of the weights and thresholds of the network, it was decided to use the VGG16 network, already trained on a million images (1000 images per category) from the ImageNet training set [9]. The VGG16 model is part of the Keras framework, and the capabilities of this library allow you to modernize the network for your tasks: a fully connected classifier was chosen, which has only one hidden layer of 256 neurons. The RMSprop method was chosen as the optimization method. The initial learning rate was set to $2 \cdot 10^{-5}$. The size of the minibatch was five. The fully connected classifier was trained over 40 epochs, after which the overfitting effect began to be observed [1, 9].

After training the network, the classification accuracy of images on the validation set reached 92% (69 images out of 75 validation ones). The change in the percentage of correctly recognized images during training is shown in Figure 1.

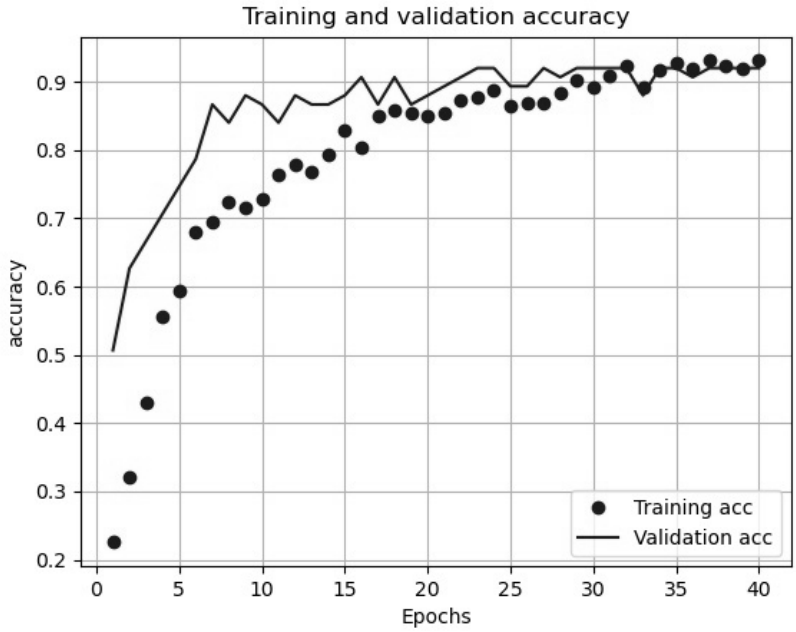


Figure 1: The process of changing the proportion of correctly recognized images in the learning process

Testing on a test set (50 images) showed that the accuracy of the trained network is 90% (45 images out of 50 test ones).

The figures in Figure 2 - Figure 6 shows examples of channels [9] of the VGG16 network, which are activated on the most characteristic signs of defects.

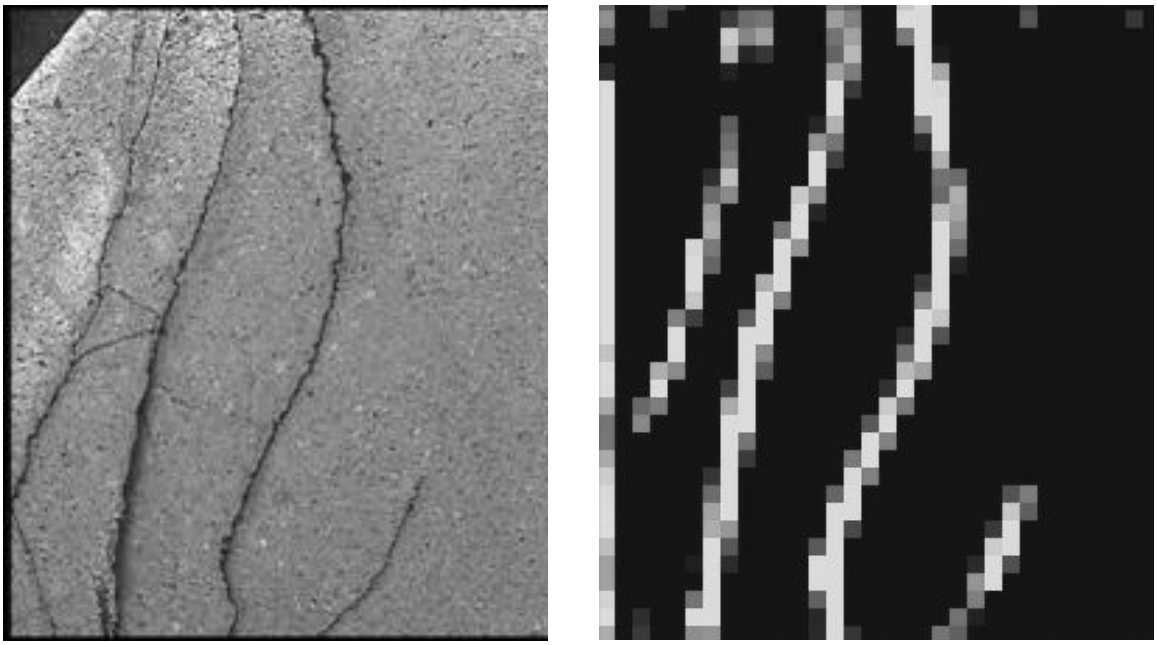


Figure 2: Activation on broken lines, which are a sign of a crack (channel 188 in the third convolutional layer of the third block of the VGG16 network)

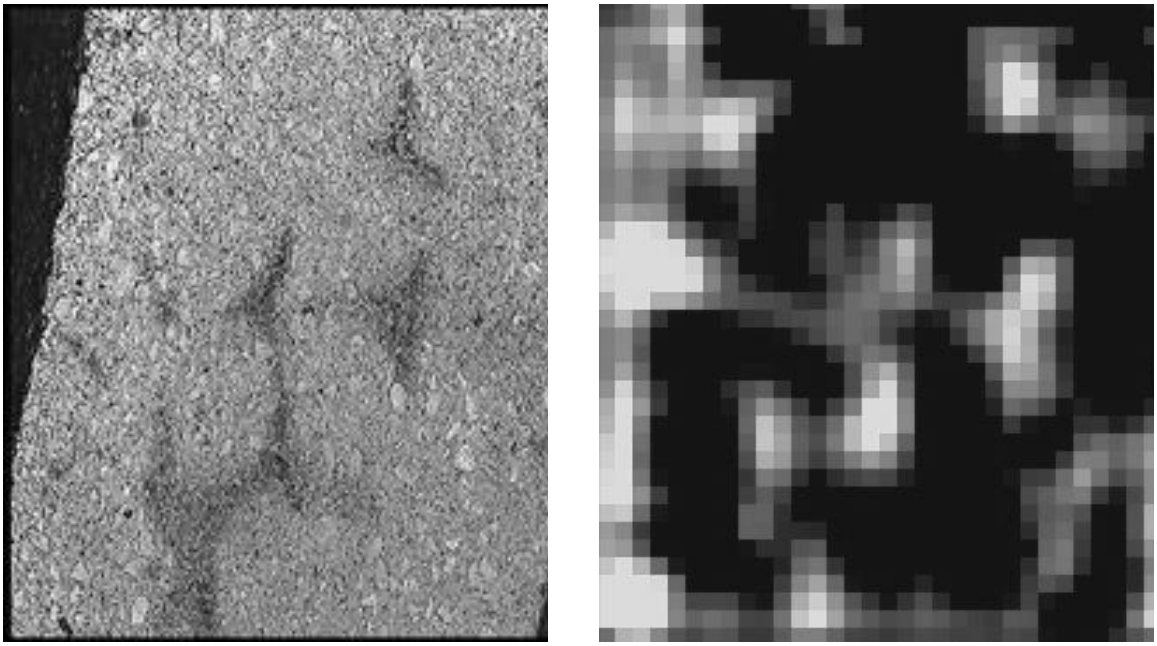


Figure 3: Activation on blackouts, which are a sign of multi hole (channel 61 in the third convolutional layer of the third block of the VGG16 network)

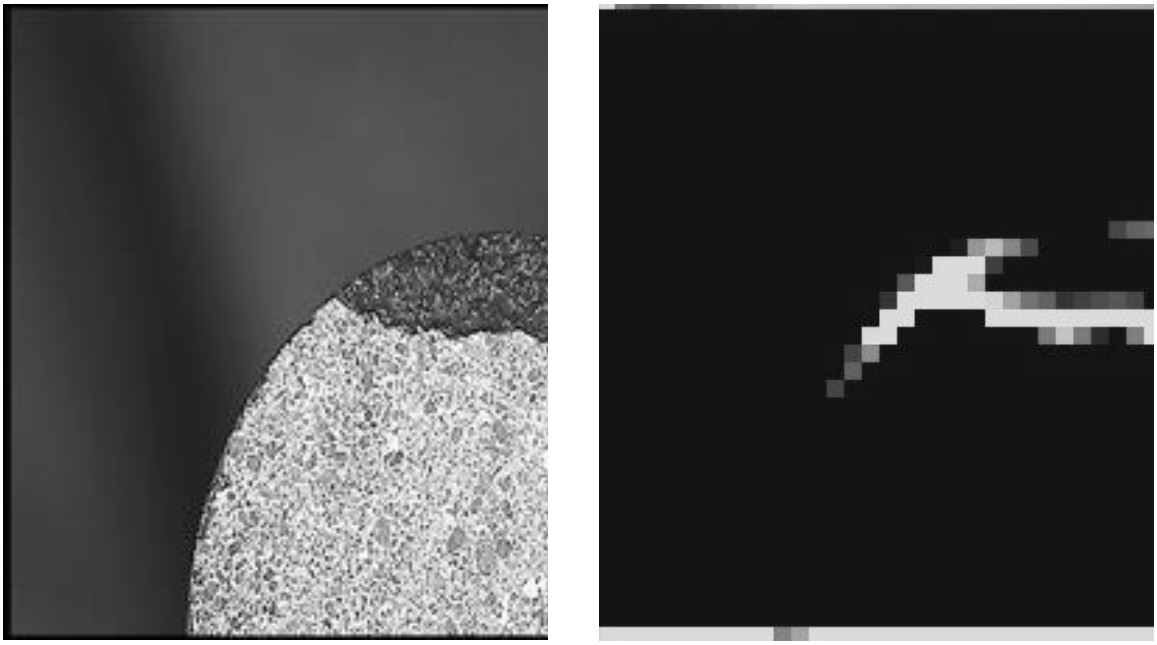


Figure 4: Activation in areas where the smooth geometry at the edges is broken, which indicates the presence of a chip (channel 182 in the third convolutional layer of the third block of the VGG16 network)

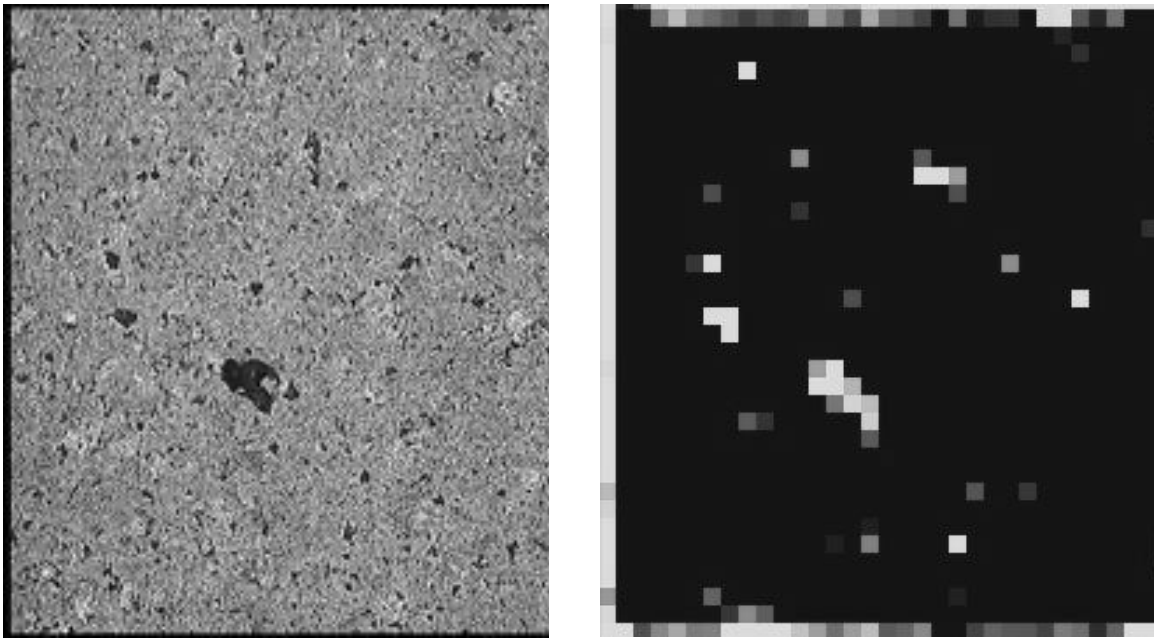


Figure 5: Activation on single spots, which are a sign of separate holes (channel 192 in the second convolutional layer of the third block of the VGG16 network)

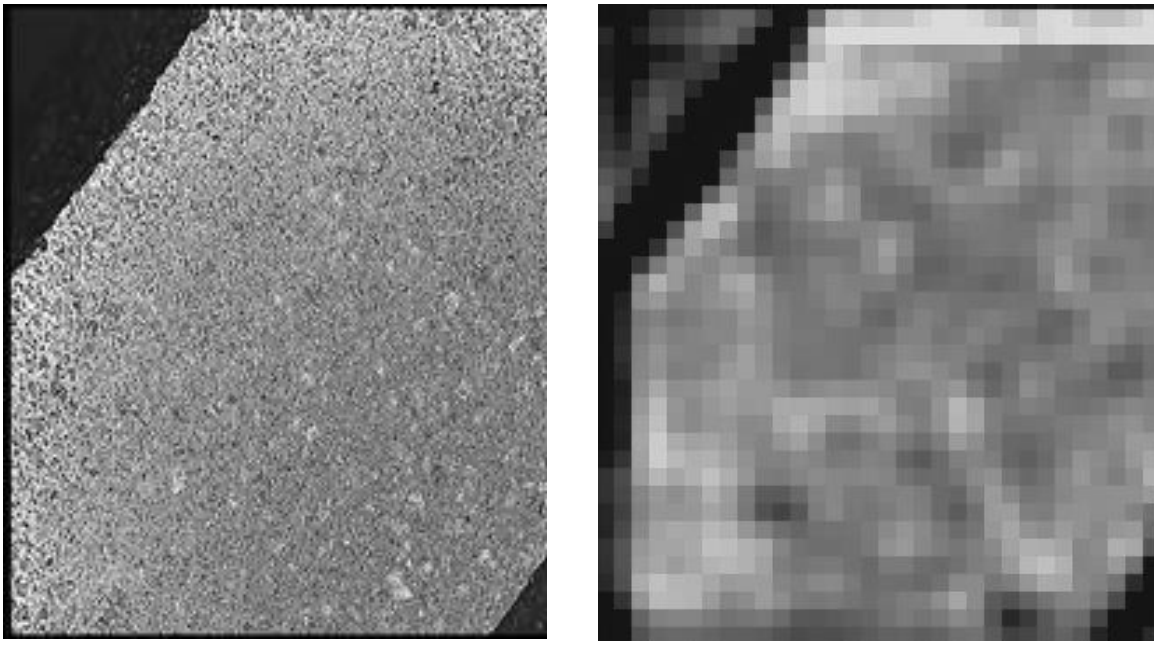


Figure 6: Activation in large areas with a uniform texture, which is a sign of a defect-free surface (channel 240 in the third convolutional layer of the third block of the VGG16 network)

3. Analysis of the results of the trained network

Examples of recognized defects are shown in Figure 7 (the most difficult cases are selected). The parentheses in the figures indicate the true categories (class labels), without parentheses, the labels predicted by the network are indicated. The percentages in the figures are the predictive probability [1, 9] of class membership.

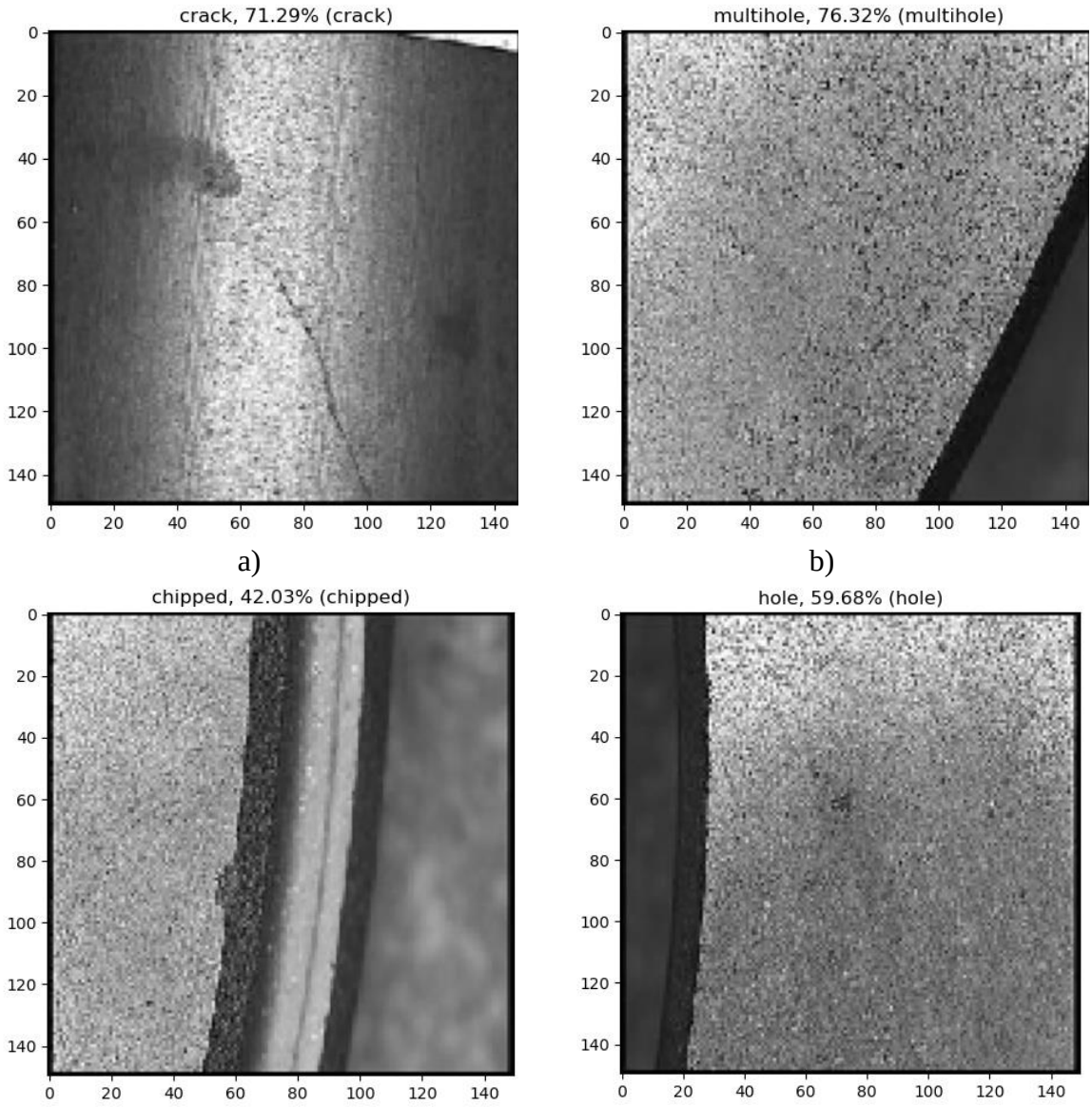
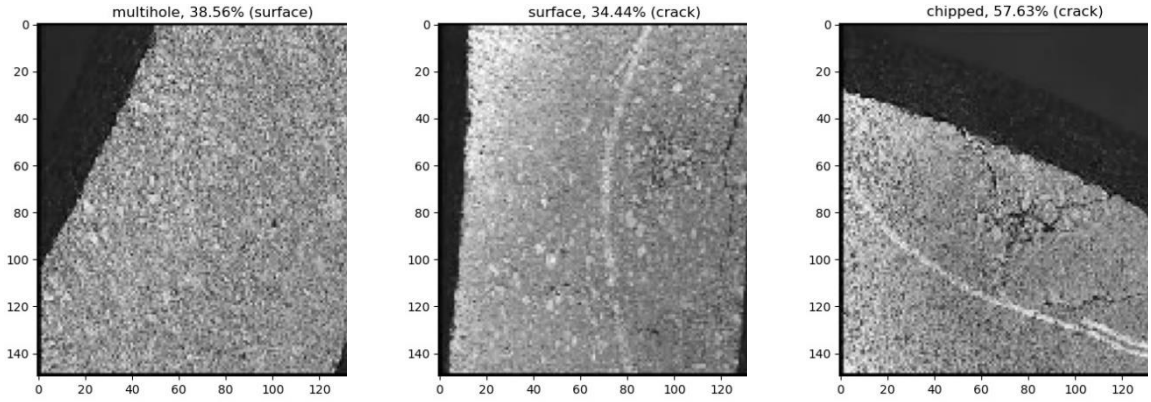


Figure 7: Examples of the trained network operation: a – «crack» type defect; b - defect of the «multi hole» type; c - defect of the «chip» type; d - defect of the «separate hole» type

Of particular interest are the faulty verdicts handed down by the network. In a test set of 50 images, only 5 images were incorrectly identified. In the validation set of 75 images, 6 images were incorrectly identified (at the time of the end of training). The two images are poorly classified even by an experienced operator. Another six out of eleven incorrectly recognized images are “uncharacteristic” images for the training sample. Such images should not be presented to the network: the use of image data in network testing is due to the limited availability of test and validation images. Another option for solving the problem is additional training or retraining of the network using "problem" images.

Separately, it is necessary to pay attention to three erroneous verdicts of the network, which, upon first examination, may seem rather rude. In Figure 8a shows a defect-free surface that has been classified by the network as a "pore pool". In Figure 8b shows a defect of the “crack” type (the network classified the defective product as a defect-free surface). Nevertheless, the predictive probability for the cases presented in the figures in Figure 8a and Figure 8b is rather low, that is, the network “doubts” its verdict. In Figure 8c shows a clearly visible crack at the edge of the product. However, the network with a probability of 57.63% passed the verdict that the image shows a chip. This problem can be solved by adding images of products with cracks closed to the edges of the product to the training set.



a) b) c)

Figure 8: Errors made by the network

4. Recommendations for using a trained neural network

To formulate recommendations for using the network, you must first of all proceed from the analysis of errors made by the network. Erroneously recognized defects were found only among images with a predictive probability of less than 60% (Figure 8c shows an incorrectly recognized defect, the predictive probability for which is the highest among all incorrectly recognized images). Thus, if an engineer is interested not only in the fact of a defect, but also in its type, then any verdict made by the network with a probability of less than 60% should be considered doubtful. Such "questionable" images must be sent to an experienced professional for a final decision.

On the contrary, all images for which the predictive probability exceeded 60% were correctly recognized by the network. This fact allows us to make a rather rough assumption that all verdicts for which the predictive probability exceeds 60% are reliable. Such an assumption, in spite of all its roughness, is quite acceptable for control processes, in which a certain percentage of errors is pre-built.

Among all the test and validation images (a total of 125 images), there were only 21 images for which the network delivered a verdict with a predictive probability of less than 60%. That is, a rough estimate based on the analysis of the recognition results of test and validation samples shows that a trained neural network saves 83% (104 images from 125) time for examining samples from silicified graphite. That is, if the operator does not check for image defects, the predictive probability for which exceeds 60%, he will save 83% of the working time.

If the researcher is interested only in the fact of the presence of a defect, and not in its type, then the percentage of "doubtful" images should be estimated from Figure 8b. The fact is that among all test and validation images with defects, only one image (Figure 8b) was incorrectly classified as a defect-free surface. Such errors are the most dangerous, since the defect that the network "overlooked" can be harmful. Based on Figure 8b, then we can conclude that all images for which the network delivered a verdict with a predictive probability of less than 40% should be sent to an experienced specialist for additional research. Among all test and validation images, only 6 images were identified that meet this requirement. Thus, if the researcher is only interested in the fact of the presence of a defect on the surface, then the trained network will save 95% of the time spent on examining samples.

5. Conclusions

Taking into account the smallness of the training set (when training commercial networks, 1000 images per category are used), the result obtained (90% of correctly recognized images on the test set) indicates the effectiveness of using a pre-trained neural network of a simple architecture. This effect is most likely due to the fact that defects do not need to reveal any abstract features. That is, to recognize objects that do not have high-level features, it is quite sufficient to use pre-trained networks with a

simple architecture. Only a fully connected classifier will be trained, which will significantly save time for training.

6. References

[1] E. Arkhangelskaya, Deep Learning. Immersion in the world of neural networks, SPb, Peter, St. Peterburg, 2020.

[2] Y.-L. Boureau, F. Bach, Y. LeCun, J. Ponce, Learning mid-level features for recognition, in: Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR2010, San Francisco, CA, 2010, pp. 2559–2566. doi: 10.1109/CVPR.2010.5539963.

[3] D. Scherer, A. Muller, S. Behnke, Evaluation of pooling operations in convolutional architectures for object recognition, in: K. Diamantaras, W. Duch, L.S. Iliadis (Eds.), Proceedings of the 20th International Conference Artificial Neural Networks – ICANN 2010, Proceedings, Part III, Thessaloniki, Greece, 2010, pp. 92-101. doi: 10.1007/978-3-642-15825-4_10.

[4] Y. L. Boureau, J. Ponce, Y. Lecun, A theoretical analysis of feature pooling in visual recognition, in: Proceedings of the 27th International Conference on Machine Learning, ICML 2010 - Proceedings, Haifa, Israel, 2010, pp. 111-118.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015. doi: 10.1109/CVPR.2015.7298594.

[6] M. Lin, Q. Chen, S. Yan, Network in Network, 2014. 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, 14-16 April 2014. URL: http://arxiv.org/abs/1312.4400

[7] K. He, X. Zhang, S. Ren J. Sun, Deep Residual Learning for Image Recognition, in: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[8] K. He, X. Zhang, S. Ren, J. Sun, Identity Mappings in Deep Residual Networks, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol. 9908. Springer, Cham. https://doi.org/10.1007/978-3-319-46493-0_38

[9] F. Scholle, Deep Learning in Python, SPb, Peter, St. Peterburg, 2020.

[10] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al. Recent advances in convolutional neural networks, 2015. arXiv 2015, URL: http: arXiv:1512.07108.

[11] S. Srinivas, R. K. Sarvadevabhatla, K. R. Mopuri, N. Prabhu, S. S. Kruthiventi and R. V. Babu, A taxonomy of deep convolutional neural nets for computer vision, Frontiers Robot. AI, vol. 2, p.36, Jan. 2016. doi: 10.3389/frobt.2015.00036.

[12] A. Krizhevsky, I. Sutskever, G. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25 (NIPS'2012). URL: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.299.205. doi:10.1145/3065386.

[13] Keras RMSprop algorithm parameters, 2021. URL: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/RMSprop?hl=ru.

[14] Y. LeCun, L. Bottou, G. Orr, K. Muller, Efficient BackProp, in: G. Orr and M. K., (Eds.), Neural Networks: Tricks of the trade, volume 1524 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 1998, pp. 9–50. doi: 10.1007/3-540-49430-8.

[15] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, Journal of Machine Learning Research, volume 9, January 2010, pp. 249-256. URL: http://proceedings.mlr.press/v9/glorot10a.html9.

[16] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: B. Francis, D. Blei (Eds.), Proceedings of the 32nd International Conference on International Conference on Machine Learning, volume 37 of ICML'15, JMLR.org, Lille France, 2015, pp. 448–456. doi:10.5555/3045118.

[17] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014. arXiv preprint arXiv:1409.1556, 2014. URL: https://arxiv.org/abs/1409.1556

[18] K. Greff, R. K. Srivastava, J. Schmidhuber, Training Very Deep Networks. Advances in Neural Information, in Processing Systems (NIPS) 28th NIPS, Cambridge, MA, USA: MIT Press, 2015, pp. 2377–2385, doi: 10.5555/2969442.2969505.