Tiny CNN for feature point description for document analysis: approach and dataset

A. Sheshkus [1,2,3], A. Chirvonaya [3,4], V.L. Arlazarov [2,3]
[1] Moscow Institute for Physics and Technology, 141701, Russia, Moscow Region, Dolgoprudny, Institutskiy per., 9;
[2] Institute for Systems Analysis, Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, 117312, Moscow, Russia, pr. 60-letiya Oktyabrya, 9;
[3] Smart Engines Service LLC, 117312, Moscow, Russia, pr. 60-letiya Oktyabrya, 9;
[4] National University of Science and Technology "MISIS", 119049, Moscow, Russia, Leninskiy prospect, 4

Abstract

In this paper, we study the problem of feature points description in the context of document analysis and template matching. Our study shows that specific training data is required for the task especially if we are to train a lightweight neural network that will be usable on devices with limited computational resources. In this paper, we construct and provide a dataset of photo and synthetically generated images and a method of training patches generation from it. We prove the effectiveness of this data by training a lightweight neural network and show how it performs in both general and documents patches matching. The training was done on the provided dataset in comparison with HPatches training dataset and for the testing, we solve HPatches testing framework tasks and template matching task on two publicly available datasets with various documents pictured on complex backgrounds: MIDV-500 and MIDV-2019.

Keywords: feature points description, metrics learning, training dataset.

Citation: Sheshkus A, Chirvonaya A, Arlazarov VL. Tiny CNN for feature point description for document analysis: approach and dataset. Computer Optics 2022; 46(3): 429-435. DOI: 10.18287/2412-6179-CO-1016.

Acknowledgements: This work was supported by the Russian Foundation for Basic Research (projects 18-29-26033 and 19-29-09064).

Introduction

The image description is a very important part of computer vision in modern science. The algorithms that somehow build a representation for the object are required in many scopes from image tagging and annotation for medical [1] or other purposes [2] to face verification [3]. The purpose of these methods is to transform an object (image, image patch, signal) into a vector of values. The essential property of these methods is to yield comparable vectors: the distance between these vectors must be small for the representations of the same / similar object and big for the representations of the different objects. In the scope of the document understanding and recognition these algorithms are also playing an important role. They are used for document template matching [4], forensics checks [5] and even for the recognition of the characters [6].

Two main types of descriptors are used: binary and floating point. The main advantage of the binary vectors is that the distance between them can be calculated much faster, for example, Hamming distance between binary vectors can be calculated using several hardware instructions. Another advantage is compactness: to store each value one needs only one bit. Unfortunately, there are popular algorithms (like SIFT and SURF) that provide floating point values and therefore cannot be used directly. However, floating point vectors are also used along with binary ones. While the comparison takes more time and storage consumes more space this type of descriptors is still viable because it allows us to employ more algorithms for example neural networks which naturally produce floating point values. Obtaining a neural network binary descriptor is possible but requires additional effort [7], [8] and is a separate problem.

In paper [9] authors show that published results on different descriptors comparison are inconsistent. This indicates the presence of ambiguity in the task of image description, as the different descriptors take into consideration different image characteristics. But the problem here is even more complex because similar methods and algorithms are used for various tasks. For example, in [10] authors train an image to vector neural network that clusters objects representations by their class. It is important that even though the idea of this kind of neural network is the same as for the feature point description, the meaning is completely different. In the case of features, the result must depend on the similarity of the images regardless of the type of the pictured object, meanwhile in the case of classification two images of a plane should be transformed into close vectors (fig. 1). In addition, the "image description" problem also exists in even more general form [11].

Existing datasets for the descriptors training are mostly focused on outdoors pictures (for example [12]) and / or too complex to be used for lightweight neural networks training. Moreover, these datasets contain distortions which make them ineffective for document fea-

ture point description. To solve this problem, we will introduce a new training dataset that is suitable for document feature points descriptor training but can also be used for multiple purposes. In our experiments, we will show that a very lightweight neural network trained on this dataset can show competitive results on both documents and general image patches.

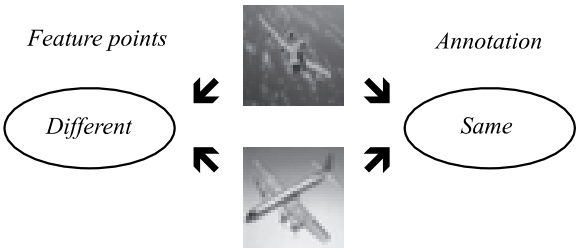


Fig. 1. Two pictures of planes. They are the same objects but completely different images

Neural networks that map input objects to a metric space are typically called metric neural networks. To train such a network several loss functions are used. One of them is a triplet loss which was known for a long time already [13]. Despite the fact that this loss function is widely used authors of many papers introduce modifications of the triplet loss for different purposes [14]. Some of them went further for a quadruple loss [15].

To summarise, in this paper we introduce a new training dataset and show how to use it to train a lightweight universal neural network descriptor. The dataset and a method for training data retrieval are provided for public usage.

1. Training dataset

1.1. Dataset creation

The training dataset consists of five parts which are collected with three different methods: synthetic generation, capturing with a camera, and direct patch generation.

In document matching, templates are usually matched by feature points descriptors. These points are often located on the static texts. So, the final descriptor must evaluate image patches with different letters as different even if these letters are located at the same places. That is why the first part of the dataset contains images with text lines synthetically generated using the method described in [16]. This approach allows printing text into the manually selected backgrounds.

The second part of the dataset aims for general purposes and consists of various textures collected from the walls, and various random surfaces with a 3D texture. These texture images introduce various shapes and their shades. To ensure the difference between the images we fix the camera and vary light source position like it is demonstrated in fig. 2.

The third part consists of the patches that were generated directly. They are blurry images with intensity peaks in random locations. To achieve this, we generated pictures with a white background and several black dots, then apply Gaussian blur and Fast Hough Transform [17] as shown in fig. 3.

Fig. 2. Process of the gathering of the second part of the dataset

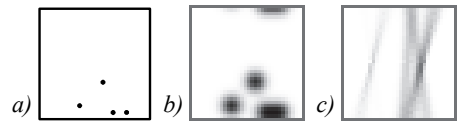


Fig. 3. Patch creation with FHT: initial points, blurred, FHT image

In contrast with the rest of the dataset, these images must introduce shapes that are not usually presented in text strings or in the wild. Still, these images are perfectly valid for a patch matching task and therefore a reasonable amount of them will increase the quality of the trained algorithm in general.

The next part is similar to the text strings but instead of letters we used hieroglyphs. This part should cover a big variety of small objects which are not presented in the standard set of symbols.

The final part of the dataset contains images of barcodes instead of letters as they have a lot of small details that the trained algorithm is expected to differentiate.

All these parts of the dataset together contain 85 images of sizes from 1150×388 to 2000×6048. Most of them were duplicated and processed with a graphical editor (embossing, gamma correction, patterning, blurring, etc.). By doing this we ensure geometrical matching between the images with the same content and introduce some visual effects which the final descriptor must tolerate. Images and their duplicates form input groups for patches retriever.

The dataset is available at *ftp://smartengines.com/desc_data*.

1.2. Patches retrieval

To create a final training set of the patches we process the dataset in a special way (scripts for patches retrieval will be available along with the dataset). We convert pictures to grayscale and retrieve image patches of the size 32×32 from all possible positions with small overlapping. This allows us to increase the number of classes and does not mix up classes. Here each class contains patches retrieved from the one position in the images of one input group. To diversify our data and extend the number of classes in the final training data we perform additional steps: add different scales of input images and patches rotations. Also, we inverse some of the images to further extend the variety of the classes. The exact values of these parameters can be found in the retrieval script.

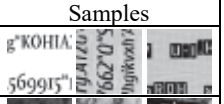
Final training data contains 325176 classes and 540716 patches. The distribution of the images per class is shown in tab. 1. It is by design that there are many single images per class as we will later employ special data augmentation.

Tab. 1. Patches per class distribution

Patches per class	1	2	3	4
Classes	149164	138436	35624	1952

The dataset parts and the number of classes they yield are summarised in tab. 2.

Tab. 2. Classes per data sources

Source	Classes number	Samples
Text lines	265384	
Photos	38916	
FHT images	10000	
Hieroglyphs	7140	
Barcodes	3736	

Since the designed dataset is created mostly for feature point description on the documents the text lines part is the biggest one in our experimental setup, but it can be balanced using the provided source code.

2. Neural network

2.1. Architecture

The neural network architecture was created with an extremely small number of trainable parameters, that is why in this work it was called “tiny”. In this architecture, layers 5 and 6 reduce dimensionality. This idea is presented in different forms in autoencoders [18], SqueezeNet [19], MobileNets [20] and others. Other than that, the neural network is quite simple: the first layer has a 4×4 window size to obtain a noticeable initial receptive field and after that the extracted features are gradually transformed into the final vector with convolutional and fully connected (FC) layers. In tab. 3 we explain the architecture details. In this neural network architecture, we use ReLU based activation function:

$$symrelu[a] = \max(-a, \min(a, x)), \quad (1)$$

where $a > 0$. This will later allow us to evaluate output value bounds. The resulting neural network has only 3.9×10^4 trainable parameters which is considered to be very small. For example, HardNet [21] neural networks have much more than 10^6 parameters. Only 2.4×10^5 summations and 2.5×10^5 multiplications are required to evaluate the result which makes this neural network suitable for usage on the device with low computations power such as various smartphones, unmanned vehicles, and others.

Tab. 3: Neural network architecture. Input shape is 32×32×1

#	Layer	Parameters	Activation	Output shape
1	Conv	8 filters 4×4, stride 2×2, no padding	symrelu[1]	$15 \times 15 \times 8$
2	Conv	8 filters 3×1, stride 1×1, no padding	symrelu[1]	$13 \times 15 \times 8$
3	Conv	8 filters 1×2, stride 1×1, no padding	symrelu[1]	$13 \times 13 \times 8$
4	Conv	20 filters 3×3, stride 2×2, no padding	symrelu[1]	$6 \times 6 \times 20$
5	Conv	16 filters 1×1, stride 1×1, no padding	symrelu[1]	$6 \times 6 \times 16$
6	Conv	12 filters 1×1, stride 1×1, no padding	symrelu[1]	$6 \times 6 \times 12$
7	Conv	20 filters 2×2, stride 1×1, no padding	symrelu[1]	$5 \times 5 \times 20$
8	Conv	48 filters 3×3, stride 2×2, no padding	symrelu[1]	$2 \times 2 \times 48$
9	FC	128 outputs	symrelu[1]	$1 \times 1 \times 128$
10	FC	16 outputs	–	$1 \times 1 \times 16$

2.2. Training

2.2.1. Batch generation

To train our neural network we used the patches from a dataset. To generate a training batch we randomly choose 8192 of them. After that, for every patch we also randomly choose one random positive example (i.e. from the same class, if there was only one patch in this class we took the same patch) and one random negative example (i.e. from a different random class). While the current batch is processed by the training framework on GPU we generate the next one on CPU. This part can be improved with various triplet generation techniques like hard mining [22, 23], but this is not a topic of the current paper therefore we use the simplest random selection.

2.2.2. Augmentation

Since our dataset does not have (and was not designed to, see tab. 1) many patches for every class, the augmentation part is essential. In our experiments, we used an online augmentation system [24]. Image distortions were different for anchor / positive elements and negative elements of the triplet (fig. 4). For anchor / positive element we carefully select the transformations which should not make the initially similar patches different: monotonic brightness changes, blur, additive noise, random crop and scale, motion blur. For the negative elements, the list of applied transformations was extended with opening and closing morphology operations, grid addition, and highlights.

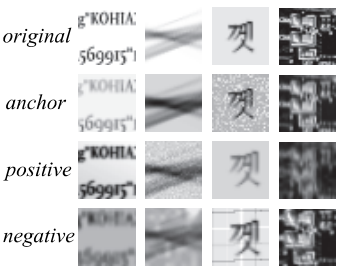


Fig. 4. Examples of images and their augmented variants

The initial probability of the image augmentation was 0.95. We select a random transformation from the list, apply it to the image with the current probability, then multiply the probability by a factor of 0.85 and repeat the procedure until the list is empty. The probability reduction is needed to prevent the data over augmentation. In other words, for every image the transformations {T} are shuffled and applied with a probability

$$p(t_i) = 0.95 \times 0.85^i. \quad (2)$$

Experiments

To prove the effectiveness of our dataset and method we performed four experiments. Firstly, we train a neural network on the HPatches training data [9] and our training data with the original triplet loss function. All neural networks were trained for approximately 5000 batches (each of which consisted of 8192 triplets) with described augmentation (see Fig. 5 for a convergence plot). For initial randomization, we use the Xavier method [25]. All the neural networks were trained with a standard triplet loss function with $\alpha = 1.5$. The convergence plot in Fig. 5 demonstrates interesting behaviour. The loss is decreasing in the training process as it should. q0 means the part of the triplets which were considered to be solved i.e. provided a zero gradient. q1 shows the part of the triplets where the distance between the anchor and the positive elements was less than $\alpha/2$. We can see that the number of such triplets is decreasing that implies that the representations of the same class grow bigger with time.

For testing purposes, we used three datasets: HPatches to check the resulting descriptor validity in general and two open datasets containing documents MIDV-500 [26] and MIDV-2019 [27] to estimate the effectiveness of the descriptor in the template matching task.

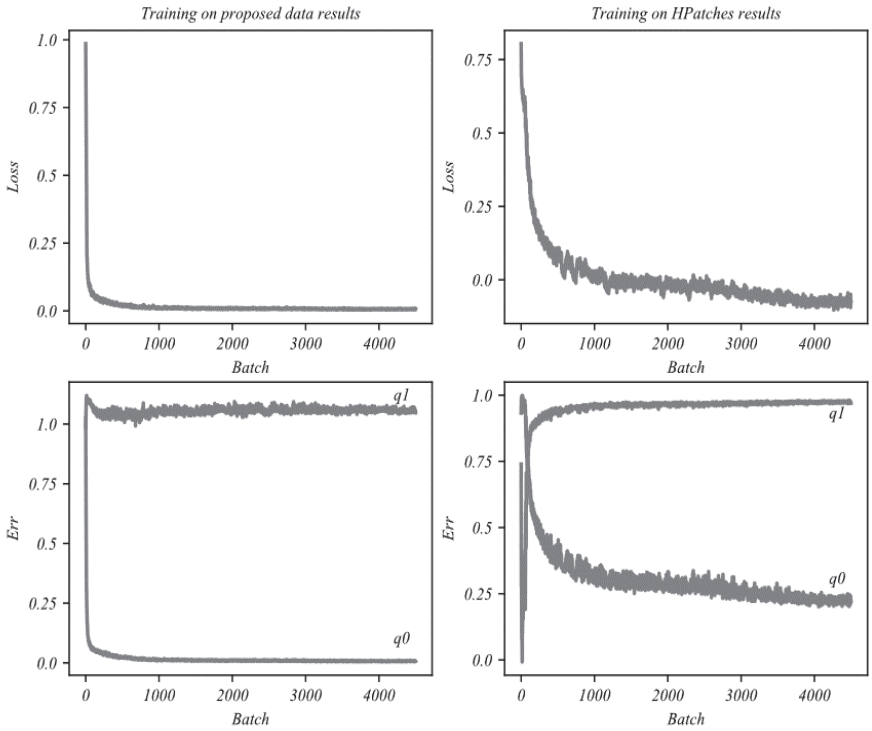


Fig. 5. Neural network convergence plot and training statistics

In fig. 6 we show some images from the used datasets. While HPatches is a dataset of the general image patches mostly containing outdoors images the MIDV-500 and MIDV-2019 datasets contain document images. The second one introduces heavier projective distortions and is considered to be harder. Both datasets have various complex backgrounds and are challenging for the task.

3. Results

In tab. 4, 5, and 6 we show the results obtained using HPatches testing framework [9]. Tables contain average precision for each subset (depending on distortions strength: E – easy pairs, H – hard, T – tough), rows were sorted by average value. In these tables "Our HP" shows the neural network trained on HPatches dataset and "Our" shows the result of the one trained on the created dataset. It can be seen that in some cases (in bold) of patch verification (see tab. 4 and 5) our training data were even better. Tab. 4 and 5 show verification results for the balanced and imbalanced number of positive and negative pairs respectively. In the retrieval task, the situation is even better (see tab. 6). The neural network trained on our data shows comparable results.

On contrary, on the documents testing datasets we can see, that HPatches training data are not good for the task while our training dataset is suitable. The results from tab. 7 prove two main points: we need specific training data for documents feature points matching and our data is valid.

Fig. 6. Examples from the testing datasets: a)HPatches, b)MIDV-500, c)MIDV-2019

Tab. 4. Verification task results (balanced). E - easy, H - hard, T - tough

Method	E-inter	E-intra	H-inter	H-intra	T-inter	T-intra
ROOTSIFT [28]	0.904	0.874	0.799	0.762	0.715	0.680
BRIEF [29]	0.881	0.874	0.814	0.806	0.748	0.741
SIFT [30]	0.931	0.910	0.823	0.796	0.731	0.705
Our	0.929	**0.920**	0.829	0.816	0.735	0.723
Binboost [31]	0.923	0.911	0.858	0.842	0.784	0.767
DC-SIAM [32]	0.933	0.910	0.875	0.843	0.813	0.776
Our HP	0.936	0.904	0.893	0.846	0.847	0.791
DC-SIAM2STREAM [32]	0.951	0.934	0.920	0.896	0.874	0.842
Deepdesc [33]	0.959	0.936	0.931	0.896	0.888	0.842
TFEAT-MARGIN-STAR [34]	0.963	0.947	0.937	0.913	0.894	0.861
TFEAT-RATIO-STAR [34]	0.962	0.945	0.939	0.913	0.898	0.864
HARDNET [21]	0.980	0.970	0.961	0.943	0.918	0.890
HARDNET+ [21]	0.981	0.971	0.962	0.945	0.920	0.893

Tab. 5. Verification task results (imbalanced). E - easy, H - hard, T - tough

Method	E-inter	E-intra	H-inter	H-intra	T-inter	T-intra
ROOTSIFT [28]	0.778	0.695	0.582	0.484	0.455	0.367
BRIEF [29]	0.727	0.700	0.563	0.536	0.444	0.422
SIFT [30]	0.849	0.783	0.657	0.570	0.512	0.429
Binboost [31]	0.814	0.769	0.665	0.614	0.534	0.488
Our	**0.838**	**0.810**	0.650	0.615	0.501	0.469
Our HP	0.837	0.756	0.722	0.616	0.619	0.509
DC-SIAM [32]	0.845	0.785	0.724	0.644	0.609	0.524
DC-SIAM2STREAM [32]	0.884	0.841	0.806	0.745	0.705	0.632
Deepdesc [33]	0.904	0.851	0.830	0.751	0.733	0.640
TFEAT-MARGIN-STAR [34]	0.916	0.874	0.846	0.781	0.748	0.668
TFEAT-RATIO-STAR [34]	0.912	0.868	0.848	0.781	0.752	0.671
HARDNET [21]	0.955	0.929	0.909	0.864	0.819	0.754
HARDNET+ [21]	0.958	0.931	0.914	0.870	0.828	0.766

Tab. 6. Retrieve task results

Method	100	500	1000	5000	10000	15000	20000	Mean
BRIEF [29]	0.477	0.328	0.279	0.195	0.168	0.154	0.146	0.250
Our HP	0.600	0.405	0.335	0.213	0.173	0.153	0.142	0.289
Our	0.578	**0.415**	**0.356**	**0.247**	**0.210**	**0.191**	**0.180**	0.311
Binboost [31]	0.575	0.416	0.363	0.269	0.235	0.218	0.208	0.326
SIFT [30]	0.634	0.503	0.458	0.372	0.341	0.324	0.314	0.421
ROOTSIFT [28]	0.625	0.501	0.460	0.384	0.355	0.340	0.331	0.428
DC-SIAM2STREAM [32]	0.709	0.562	0.509	0.399	0.360	0.339	0.326	0.458
DC-SIAM [32]	0.726	0.575	0.521	0.410	0.370	0.349	0.335	0.469
TFEAT-RATIO-STAR [34]	0.737	0.602	0.549	0.437	0.397	0.375	0.361	0.494
TFEAT-MARGIN-STAR [34]	0.745	0.614	0.564	0.455	0.415	0.394	0.380	0.510
Deepdesc [33]	0.774	0.644	0.587	0.469	0.427	0.403	0.388	0.527
HARDNET [21]	0.860	0.772	0.736	0.653	0.620	0.601	0.589	0.690
HARDNET+ [21]	0.861	0.773	0.738	0.655	0.623	0.605	0.593	0.693

These datasets contain template images for each of the 50 document types that are used for frames matching. Location error for every frame is the maximal deviation of the computed document corner coordinates divided by the length of the shortest document boundary side. Such an error is computed for every template and the least one points to the most possible document type. In this table mean localization error for every dataset is presented.

Tab. 7. MIDV-500 and MIVD-2019 results

NN	MIDV-500	MIDV-2019
Our HP	0.857	0.941
Our	0.290	0.520

Another interesting point is that one can notice that even though for the triplet training multiple examples per class are needed to construct anchor positive occurrences in our data there are many classes with a single image. It may seem that this is a disadvantage of the dataset but in fact on contrary. With this data distribution, we can carefully choose augmentation for anchor and positive image and control which transformations should be tolerated, and which should be not.

Moreover, since most of the data is randomly generated, we cannot be 100 % sure that there are no images in different classes that are very similar. But our analysis shows that the probability of this is very low. Furthermore, with over 3×10^5 classes the chance, that two exact images will be selected incorrectly is negligible.

Finally, as we provided collected images and script for patches retrieval, there is a variety of ways to generate datasets for training using configuration parameters. For example, it is possible to control scales choosing objects' size on patches that is suitable for the task.

Conclusion

In this paper, we showed that feature points description is different for documents and outdoor images. The comparison of the trained neural networks on the general and special (ours) datasets clearly shows the gap in the quality. The main purpose of our dataset is to provide the necessary information for the description of the image patches containing letters. Additional images make the training data applicable not only for document image patches matching but for other purposes as well. We also demonstrated that a very lightweight neural network can still be used for the task which makes this kind of algorithm applicable when using on devices with limited computational resources.

For future work we plan to further enhance the dataset in two main different ways: evaluate what type of data is still missing and add new images and improve the patches retrieval mechanism to use the already presented data even more efficiently. Also, we plan to quantize the neural network and its output down from 32 bits per value to 8 bits per value which should be possible without (or with minimal) quality loss according to the neural network properties. We also plan to study the possibility of the output dimension reduction for further descriptor size shrink.

References

[1] Kougia V, Pavlopoulos J, Androutsopoulos I. Medical image tagging by deep learning and retrieval. In Book: Arampatzis A. et al, eds. Experimental IR meets multilinguality, multimodality, and interaction. CLEF 2020. Cham: Springer; 2020: 154-166. DOI: 10.1007/978-3-030-58219-7_14.

[2] Shin Y, Seo K, Ahn J, Im DH. Deep-learning-based image tagging for semantic image annotation. In Book: Park J, Park DS, Jeong YS, Pan Y, eds. Advances in computer science and ubiquitous computing. CSA-CUTE 2018. 2018. Singapore: Springer; 2019: 54-59. DOI: 10.1007/978-981-13-9341-9_10.

[3] William I, Ignatius Moses Setiadi DR, Rachmawanto EH, Santoso HA, Sari CA. Face recognition using FaceNet (survey, performance test, and comparison). Fourth Int Conf on Informatics and Computing 2019 (ICIC) 2019; 1: 1-6. DOI: 10.1109/ICIC47613.2019.8985786.

[4] Skoryukina N, Arlazarov V, Nikolaev D. Fast method of ID documents location and type identification for mobile and server application. Int Conf on Document Analysis and Recognition, 2019 (ICDAR) 2019; 1: 850-857. DOI: 10.1109/ICDAR.2019.00141.

[5] Kumar M, Gupta S, Mohan N. A computational approach for printed document forensics using SURF and ORB features. Soft Comput 2020; 24(1): 13197-13208. DOI: 10.1007/s00500-020-04733-x.

[6] Ilyuhin SA, Sheshkus AV, Arlazarov VL. Recognition of images of Korean characters using embedded networks. In Book: Wolfgang O, Nikolaev D, Zhou J, eds. Twelfth Int Conf on Machine Vision 2019 (ICMV) 2020; 11433: 1-7. DOI: 10.10007/1234567890.

[7] Duan Y, Lu J, Wang Z, Feng J, Zhou J. Learning deep binary descriptor with multi-quantization. IEEE Conf on Computer Vision and Pattern Recognition 2017; 1: 1183-1192. DOI: 10.1109/CVPR.2017.516.

[8] Zhang J, Ye S, Huang T, Rui Y. CDbin: Compact discriminative binary descriptor learned with efficient neural network. IEEE Trans Circuits Syst Video Technol 2020; 30(3): 862-874. DOI: 10.1109/TCSVT.2019.2896095.

[9] Balntas V, Lenc K, Vedaldi A, Mikolajczyk K. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. IEEE Conf on Computer Vision and Pattern Recognition 2017: 5173-5182.

[10] Hoffer E, Ailon N. Deep metric learning using triplet network. In Book: Feragen A, Pelillo M, Loog M, eds. Similarity-based pattern recognition 2015 (SIMBAD). Cham: Springer; 2015: 84-92. DOI: 10.1007/978-3-319-24261-3_7.

[11] Mishra A, Liwicki M. Using deep object features for image descriptions. arXiv preprint. Source: ⟨https://arxiv.org/abs/1902.09969⟩.

[12] Paulin M, Douze M, Harchaoui Z, Mairal J, PerroninF, Schmid C. Local convolutional features with unsupervised training for image retrieval. 2015 IEEE Int Conf on Computer Vision (ICCV) 2016; 1: 91-99. DOI: 10.1109/ICCV.2015.19.

[13] Schultz M, Joachims T. Learning a distance metric from relative comparisons. Adv Neural Inf Process Syst 2004; 16(1): 41-48.

[14] Cacheux YL, Borgne HL, Crucianu M. Modeling inter and intra-class relations in the triplet loss for zero-shot learning. Proc IEEE/CVF Int Conf on Computer Vision (ICCV) 2019; 1: 10333-10342.

[15] Chen W, Chen X, Zhang J, Huang K.: Beyond triplet loss: a deep quadruplet network for person re-identification. Proc IEEE Conf on Computer Vision and Pattern Recognition (CVPR) 2017; 1: 403-412.

[16] Chernyshova YS, Gayer AV, Sheshkus AV. Generation method of synthetic training data for mobile OCR system. Proc SPIE 2018; 10696: 106962G. DOI: 10.1117/12.2310119.

[17] Nikolaev DP, Karpenko SM, Nikolaev IP, Nikolayev PP. Hough transform: underestimated tool in the computer vision field. Proc 22th European Conf on Modelling and Simulation 2008: 238-246. DOI: 10.7148/2008-0238.

[18] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science 2006; 313(5786): 504-507. DOI: 10.1126/science.1127647.

[19] Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint. Source: ⟨https://arxiv.org/abs/1602.07360⟩.

[20] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint. Source: ⟨https://arxiv.org/abs/1704.04861⟩.

[21] Mishchuk A, Mishkin D, Radenovic F, Matas J. Working hard to know your neighbor's margins: Local descriptor learning loss. NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems 2017: 4826-4837.

[22] Zhao Y, Jin Z, Qi GJ, Lu H, Hua XS. An adversarial approach to hard triplet generation. In Book: Ferrari V, Hebert M, Sminchisescu C, Weiss Y, eds. Computer vision – Proceedings of the European conference on computer vision 2018. Cham: Springer; 2018: 501-517. DOI: 10.1007/978-3-030-01240-3_31.

[23] Sikaroudi M, Ghojogh B, Safarpoor A, Karray F, Crowley M, Tizhoosh HR. Offline versus online triplet mining based on extreme distances of histopathology patches. In Book: Bebis G. et al, eds. Advances in visual computing 2020. Cham: Springer; 2020: 333-345. DOI: 10.1007/978-3-030-64556-4_26.

[24] Gayer AV, Chernyshova YS, Sheshkus AV. Effective real-time augmentation of training dataset for the neural networks learning. Proc SPIE 2018; 11041: 10411I. DOI: 10.1117/12.2522969.

[25] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Proc Thirteenth Int Conf on Artificial Intelligence and Statistics (AISTAST) 2010; 9: 249-256.

[26] Arlazarov VV, Bulatov KB, Chernov TS, Arlazarov VL. MIDV-500: A dataset for identity document analysis and recognition on mobile devices in video stream. Computer Optics 2019; 43(5): 818-824. DOI: 10.18287/2412-6179-2019-43-5-818-824.

[27] Bulatov K, Matalov D, Arlazarov VV. MIDV-2019: challenges of the modern mobile-based document OCR. Proc SPIE 2019; 11433: 114332N. DOI: 10.1117/12.2558438.

[28] Arandjelovic R, Zisserman A. Three things everyone should know to improve object retrieval. Proc 2012 IEEE Conf on Computer Vision and Pattern Recognition 2012: 2911-2918. DOI: 10.1109/CVPR.2012.6248018.

[29] Calonder M, Lepetit V, Strecha C, Fua P. BRIEF: Binary robust independent elementary features. In Book: Daniilidis K, Maragos P, Paragios N, eds. Proceedings of the 11th European conference on computer vision. Berlin, Heidelberg: Springer; 2010: 778-792. DOI: 10.1007/978-3-642-15561-1_56.

[30] Lowe DG. Object recognition from local scale-invariant features. Proc Seventh IEEE Int Conf on Computer Vision 1999; 2: 1150-1157. DOI: 10.1109/ICCV.1999.790410.

[31] Trzcinski T, Christoudias M, Lepetit V. Learning image descriptors with boosting. IEEE Trans Pattern Anal Mach Intell 2015; 37(3): 597-610. DOI: 10.1109/TPAMI.2014.2343961.

[32] Zagoruyko S, Komodakis N. Learning to compare image patches via convolutional neural networks. Proc 2015 IEEE Conf on Computer Vision and Pattern Recognition (CVPR) 2015: 4353-4361. DOI: 10.1109/CVPR.2015.7299064.

[33] Simo-Serra E, Trulls E, Ferraz L, Kokkinos I, Fua P, Moreno-Noguer F. Discriminative learning of deep convolutional feature point descriptors. Proc 2015 IEEE Int Conf on Computer Vision (ICCV) 2015: 118-126. DOI: 10.1109/ICCV.2015.22.

[34] Balntas V, Riba E, Ponsa D, Mikolajczyk K. Learning local feature descriptors with triplets and shallow convolutional neural networks. Proc British Machine Vision Conf 2016: 119.1-119.11. DOI: 10.5244/C.30.119.

Authors' information

Alexander Vladimirovich Sheshkus (b. 1986) received the B.S. and M.S. degrees in Applied Physics and Mathematics from Moscow Institute of Physics and Technology (State University), Moscow, Russia, in 2009 and 2011, respectively. He is currently the head of Machine Learning department in Smart Engines, and a researcher in FRC "Computer Science and Control" of RAS. His research interests include deep neural networks, computer vision and projective invariant image segmentation. E-mail: *astdcall@gmail.com*.

Anastasiya Nikolaevna Chirvonaya (b. 1998) received the B.S. degree in Applied Mathematics from National University of Science and Technology "MISIS", Moscow, Russia, in 2019. She is currently pursuing the M.S. degree in Applied Science at the same university. She works as a programmer at Smart Engines. Her research interests are computer vision and machine learning. E-mail: *nastyachirvonaya@smartengines.com*.

Vladimir Lvovich Arlazarov (b. 1939) Dr. Sc., corresponding member of the Russian Academy of Sciences, graduated from Lomonosov Moscow State University in 1961. Currently he works as head of sector 9 at the Institute for Systems Analysis FRC "Computer Science and Control" RAS. Research interests are game theory and pattern recognition. E-mail: *vladimir.arlazarov@gmail.com*.

Code of State Categories Scientific and Technical Information 28.23.15
Received July 23, 2021. The final version – October 22, 2021.