Methods of information systems protection

UDC 004.934:141.1 **doi:** https://doi.org/10.20998/2522-9052.2023.2.12

Serhii Datsenko, Heorhii Kuchuk

National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine

BIOMETRIC AUTHENTICATION UTILIZING CONVOLUTIONAL NEURAL NETWORKS

Abstract. Relevance. Cryptographic algorithms and protocols are important tools in modern cybersecurity. They are used in various applications, from simple software for encrypting computer information to complex information and telecommunications systems that implement various electronic trust services. Developing complete biometric cryptographic systems will allow using personal biometric data as a unique secret parameter instead of needing to remember cryptographic keys or using additional authentication devices. **The object of research** the process of generating cryptographic keys from biometric images of a person's face with the implementation of fuzzy extractors. **The subject of the research** is the means and methods of building a neural network using modern technologies. **The purpose of this paper** to study new methods for generating cryptographic keys from biometric images using convolutional neural networks and histogram of oriented gradients. **Research results.** The proposed technology allows for the implementation of a new cryptographic mechanism - a technology for generating reliable cryptographic passwords from biometric images for further use as attributes for access to secure systems, as well as a source of keys for existing cryptographic algorithms.

Keywords: biometric cryptographic systems; cryptographic keys; fuzzy extractors; convolutional neural network.

Introduction

Relevance. In contemporary cybersecurity, cryptographic algorithms and protocols are crucial tools. They are applied in a variety of applications, from straightforward computer information encryption software to sophisticated communications and information systems that incorporate multiple electronic trust services. Implementing comprehensive biometric cryptography systems will enable the use of personal biometric data as a unique secret parameter rather than having to remember cryptographic keys or rely on extra authentication devices.

Interest in biometric methods has grown drastically in recent years. Modern technologies replace traditional biometric systems by forming cryptographic keys on the spot, as discovered by comparing acquired biometric photos with preserved reference copies [1 - 4]. The development of full-fledged biometric cryptography systems, in which biometric data of personality should be applied as a source of unique secret parameters, might be the next step in the advancement of such technology [5, 6]. The end user no longer has to remember cryptographic keys (passwords) or utilize extra devices to transmit, store, and etc. The biometric cryptosystem may be initialized at anytime and anywhere by removing the required parameters on the spot (with practicable erasures, mistakes, etc.) without causing harm to the given pictures [5, 7].

An overview of scientific works. The process of authentication involves employing several identifying measures to verify the user's validity [8 - 10]. In the security system, during authentication procedure the information provided by the user will be compared with the database and upon the match, user will be granted access to this system [11, 12]. For user identification, biometric authentication systems rely on their distinctive traits [4, 5, [13]. Process whereby the person is automatically identified based on a vector of characteristics selected from their physiological or behavioral features [5, 7]. This leads to a classification of biometric approaches into two categories: physiological and behavioral [4, 13, 14]. Physical characteristics that a person already holds, such as their hand, fingerprint, or face, are used in physiological biometrics. This usually originates from the fact that a person's features remain constant over time. Behavioral biometrics, on the other hand, are based on the user's actions, such as how they take notes or write articles [15].

Setting objectives. Analysis of biometric features involves a variety of research methods. Convolutional neural networks (CNN) are the most widespread. Histogram of oriented graphs (HOG) is another mathematical tool for pattern identification in computer vision systems. Based on this, **the goal of this paper** is to study these methods, their software implementation and experimental researches of their performance to solve problems of biometric authentication. In particular, the authentication precision and biometric image processing speed of CNN and HOG are evaluated.

Software implementation and convolutional neural network model description

The Python programming language utilizing face_recognition and dlib modules was used to develop software implementation of authentication algorithms for biometric images of facial features. These modules provide functions for HOG and CNN technologies, as well as the choice between 2 models (small standard and larger one) to read additional biometric features. The program allows us to detect facial features and compare them to those that remained in the collection.

A. Applied libraries and functions

The following libraries were used during software development.

1) Dlib - a modern C++ toolkit that incorporates machine learning techniques and tools for constructing multiplex software to solve real-world problems.

2) Mathematical library NumPy, which offers the following functionalities:

– powerful N-dimensional arrays;
– tools for integrating C/C++ and Fortran code;
– useful features of linear algebra, Fourier transformations and random numbers;
– complex (notification) functions;
– and much more.

3) The face_recognition library has been generated with the most recent dlib face recognition library. Its functionality enables us to recognize and manage faces. The Face Recognition library is made up of multiple code files and models that have been trained on a large number of examples and are freely available. It's also important to mention that the model scored 99.38% on the Labeled Faces in the Wild test

4) Face_recognition_models models from the auxiliary library.

5) Pillow Library (PIL) - a set of functions that adds the ability to process images to the Python interpreter. This library supports a huge range of file types, has efficient internal representation, and has a rather broad image processing capacity. The basic set of modules is intended to provide easy access to data saved in many basic pixel formats. It should serve as a solid foundation for the image processing tool as a whole.

During the development of the test software the following functions were utilized [16]:

1) The function load_image_file, which loads pictures for analysis.

2) The face_encodings function, which gives us access to the transmitted image array's face encoding.

3) The face_locations function provides an array of bounding squares for faces on the picture after taking the image file, the type of model (algorithm) to be used and the number of sampling times.

4) The face_distance function, which calculates the Euclidean distance by comparing the original image's encoding to the encoding of another face. How similar the faces are can be determined by this distance.

5) A list of base face encodings is compared to a list of candidate face encodings using the compare_faces function.

6) The percentdif function determines the likelihood that two bit strings will match.

7) A binar function that, using the received face metrics, produces a bit string.

B. Convolutional neural network model description

CNN-based object detector is trained by implementing the loss layer, known as loss_mmod, that is provided in the dlib library. To create this layer of loss, a unique method known as Max-Margin Object Detection (MMOD) is used. HOG functions are replaced by CNN in order to produce the same loss layer as the popular SVM+HOG object detector, and after that, the entire detector will be trained from beginning to end [17]. This opens up an opportunity for the development of stronger detectors.

First step is to define our CNN. CNN will be assessed in a complicated manner across the picture pyramid, namely the standard sliding window classifier. Herewith the CNN must be identified with the intention to view a part of the image and determine whether we are looking for the right object. In order to clearly determine whether a face is present in a 50x50 image, for example, we could define a CNN that uses a receptive field and has a roughly 50x50 pixel size, which is suitable for face detection. CNN with unique architectures may be advantageous for other applications.

For instance, CNN is configured with three layers of sample reduction [17]. The image will be scaled eight times with help of these layers, presenting a useful map with 32 sizes. The outcome of the preceding stage is then passed through 4 more convoluted layers to create the final network output. When the network considers locating an object in a specific place, the values will be enormous since the remaining layer consists of only 1 channel.

Network definition starts from creating some network blocks.

To reduce the sample in two times a 5x5 conversion layer is presented:

```
template <long num_filters, typename SUBNET> using
con5d = con<num_filters,5,5,2,2,SUBNET>;
```

A 3 × 3 conversion layer without reducing sampling is presented:

```
template <long num_filters, typename SUBNET> using
con3  = con<num_filters,3,3,1,1,SUBNET>;
```

With reference to convolutional 5d blocks we can now define an 8x sampling reduction block. Use of ReLU and batch normalization in the conventional manner is also applicable:

```
template <typename SUBNET> using downsampler  =
relu<bn_con<con5d<32, relu<bn_con<con5d<32,
relu<bn_con<con5d<32,SUBNET>>>>>>>>>;
```

The remaining network transforms into 3x3 convolutional layers with batch normalization and reuse. Therefore, the 3x3 block, that is used here, is defined:

```
template  <typename  SUBNET>  using  rcon3   =
relu<bn_con<con3<32,SUBNET>>>;
```

The entire network is completely defined. A special layer known as input_rgb_image_pyramid forces the network to operate on the spatial pyramid, which keeps the detector's scale constant:

```
using net_type  =
loss_mmod<con<1,6,6,1,1,rcon3<rcon3<rcon3<downsa
mpler<input_rgb_image_pyramid<pyramid_down<6>>>>
>>>>;
```

In this scenario, the face detector will be trained using the catalog's small face data set. So, first and foremost, we must acquire this dataset:

```
const std::string faces_directory = argv[1];
```

A training data set and a separate test data set are comprised in the face catalog. The training data consists of four pictures, each of which is highlighted by rectangles that delimit each distinct human face. The goal is to use this training data to figure out how to detect people's faces in new photographs.

Furthermore, once an object detector has completed its training, it must be tested on data that has not been educated. As a result, a separate test set of five pictures is also loaded. The efficiency of the face detector derived from training data will be assessed by running it on other test photos.

As a result, variables containing a set of data are generated here. The position of training image faces will be stored in face_boxes_train, whereas images_train will contain four training images. For example, the image images_train [0] includes faces described by rectangles in the array face_boxes_train [0]:

```
std::vector<matrix<rgb_pixel>> images_train,
images_test;
std::vector<std::vector<mmod_rect>>
face_boxes_train, face_boxes_test;
```

XML files containing images from each data set as well as the positions of face borders from that point can be downloaded. Any input format can be used without a doubt if the data is stored in images_train and face_boxes_train:

```
load_image_dataset(images_train,
face_boxes_train,
faces_directory+"/training.xml");
 load_image_dataset(images_test, face_boxes_test,
        faces_directory+"/testing.xml");
```

The Max-Margin Object Detection method contains numerous parameters that can be adjusted to control how it functions. In any case, we can provide the constructor with training notes and the size of the targeted object, and it will naturally adjust itself to solve our problems. Faces, on the other hand, are still recognisable at 40x40 pixels. In most cases, we should go with the smallest size possible. In accordance with the preceding rule, the constructor, defined as mmod_options, will invariably determine the required width and height of the sliding window. It will also automatically select a fair maximum for the suppression parameters:

```
mmod_options options(face_boxes_train, 40,40);
```

If necessary, multiple sliding windows can be applied to the detector automatically. However, for these faces, only one is required.

A network and a simulator can now be built:

```
net_type net(options);
```

The loss of the MMOD necessitates a number of options.detector filters equal to windows.size(). As a result, it is established here:

```
net.subnet().layer_details().set_num_filters(opt
ions.detector_windows.size());
dnn_trainer<net_type> trainer(net);
trainer.set_learning_rate(0.1);
trainer.be_verbose();
trainer.set_synchronization_file("mmod_sync",
std::chrono::minutes(5));
trainer.set_iterations_without_progress_threshol
d(300);
```

The network must be educated at this point. 150-image miniature bundles will be used. The images can be acquired by selecting random samples from the training set:

```
std::vector<matrix<rgb_pixel>>
mini_batch_samples;
std::vector<std::vector<mmod_rect>>
mini_batch_labels;
random_cropper cropper;
cropper.set_chip_dims(200, 200);
```

Shredder requires any minimum dimensions that have been transferred to the constructor mmod_options, which is accomplished here:

```
cropper.set_min_object_size(40,40);
dlib::rand rnd;
```

The simulator will continue to operate until the rate of training becomes insignificant. It takes a long time. There is an option of randomly mixing colors, which typically helps the detector better infer new images:

```
while(trainer.get_learning_rate() >= 1e-4){
cropper(150,   images_train,   face_boxes_train,
mini_batch_samples, mini_batch_labels);
for (auto&& img : mini_batch_samples)
disturb_colors(img, rnd);
trainer.train_one_step(mini_batch_samples,
mini_batch_labels);}
```

Training flows are scheduled to end soon:

```
trainer.get_net();
```

The network has been saved to disk:

```
net.clean();
serialize("mmod_network.dat") << net;
```

When a face detector is obtained, it can now be examined. The initial operation checks it on training input, whereas the secondary operation examines it on test input. Recall, accuracy, and then average accuracy will be outputted. This should indicate that the network operates properly when learning new information:

```
cout    <<    "training     results:    "    <<
test_object_detection_function(net, images_train,
face_boxes_train) << endl;
cout    <<    "testing     results:    "    <<
test_object_detection_function(net,  images_test,
face_boxes_test) << endl;
```

C. Testing methods

The algorithm of software implementation and research methodology lies in performing the following steps:

1) A basic biometric image is introduced, and the face recognition method (CNN or HOG) is chosen for the pictures.

2) Applying the face_locations and load_image_file functions to load and locate the face in the base image. Using the face_encodings function, the found face image is processed, resulting in the creation of an array showing the distances between the face's primary points (biometric image encoding).

3) The test image is loaded (from a particular sample), processed, and encoded (as for the base image from step 2), and then stored.

4) Examining the array of distances between the base image and the test (using the compare_faces function).

5) Steps 3 and 4 are repeated in a loop for every image in the sample.

6) List of solutions (the outcome of arrays of distances being compared on all test photos) is created.

7) Experimental results (match probabilities, execution time, etc.) are outputted.

Therefore, using HOG and CNN technologies, experimental research will produce digital data from biometric images. The face_encodings function, in particular, enables us to encode a face from the resulting biometric image and generates a list of 128 real numbers that characterize various facial features. Then, a single binary number with 128-digit number is created using all of the obtained values. We used the following rule: a real number is assigned a "1" if it is more than or equal to zero, and a "0" if it is less than zero. The acquired 128-digit number can be also compared to one another and utilized as a model for a key (access password) in the future. By employing passwords created in this manner, the efficiency of biometric authentication may be evaluated. The results indicate a pretty high level of match probability. We have a requirement for authentication if, specifically, the generated passwords match the matching biometric photos by more than 80%, indicating that only one user is responsible for them.

Results of Experiment

The program was evaluated using both ways on a sample of 480 test photos of various faces in accordance with the aforementioned algorithm. It was assessed how quickly biometric features were processed and how likely it was that certain features from the original image would match those in all other images. The results were compared to one another and summarized in Table 1 and Fig. 1, respectively. Each value in the table and figure, in particular, corresponds to a different biometric image from the test. The zero (0) image is the baseline (beginning) image used to compare with all other images. The procedures used before extracting the metrics from the null file refer to the model that was used to obtain the face detector, hence they take a lot longer than the following images. The processing time is roughly the same for the remaining photos. The total processing time for all 480 test photos is displayed in the table's final column. We can see that HOG's technology is a little behind CNN's.

The probabilities of matching between biometric pictures of human faces acquired using the HOG and CNN technologies are shown in Fig. 1, respectively. Only 10 pairs of typical values from 480 test results are represented in the figure. As we can see, the outcomes produced by both technologies—HOG and CNN—are essentially the same. However, CNN typically provides a more accurate answer.

The following general conclusions can be taken from the results:

– Although the HOG method addresses the face detector a little bit faster, it receives facial metrics a lot slower. On the basis of this, CNN requires less time overall to run the authentication process;

– The use of CNN technology offers somewhat more precise face metrics, which increases the likelihood that specific biometric traits will match.

Thus, it is best to use convolutional neural networks to resolve the task of recognizing bio-metric images. With the help of this technology, we can quickly and with a high degree of probability compare face metrics extracted from any image. Additionally, this technology will perform better on a wider range of samples. The requirement to use an efficient graphics processor with CUDA cores in its architecture is the main drawback of the practical application of the created software solution.

Table 1 – **Time to Obtain Face Metrics**

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**…**	**479**	**∑**
HOG	1,89	0,04	0,04	0,34	0,75	0,22	0,08	0,08	0,08	…	0,03	21,77
CNN	1,21	0,03	0,03	0,03	0,69	0,85	0,09	0,08	0,08	…	0,03	20,88

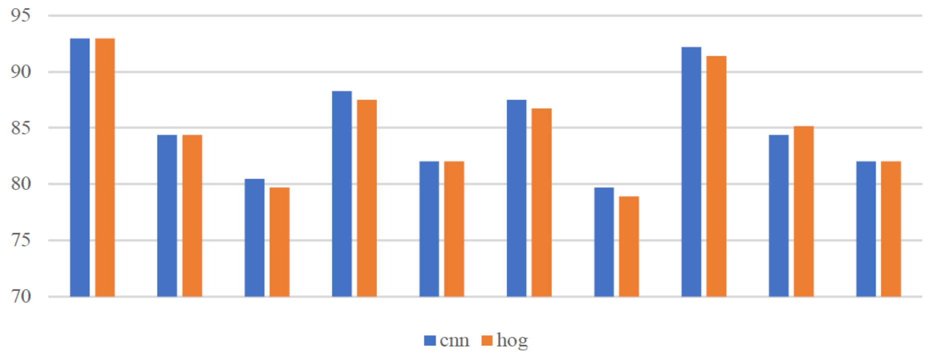

Fig. 1. Comparative diagram of the probability of match of faces

Conclusions

The following results were obtained in this paper:

1. A methods for generating cryptographic keys from biometric images using convolutional neural networks and histogram of oriented gradients was proposed.

2. Conclusions of methods comparison by probability and performance were presented.

REFERENCE

1. Dodis, Y., Reyzin, L. and Smith, A. (2004), “Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data”, *Advances in Cryptology - EUROCRYPT 2004*, Berlin, pp. 523–540, doi: https://doi.org/10.1007/978-3-540-24676-3_31.
2. Boyen, X. (2004), “Reusable cryptographic fuzzy extractors”, *Proceedings of the 11th ACM conference on Computer and communications security*, New York, NY, USA, Oct. 2004, pp. 82–91, doi: https://doi.org/10.1145/1030083.1030096.
3. Álvarez, F. H. and Encinas, L. H. (2009), “Security Efficiency Analysis of a Biometric Fuzzy Extractor for Iris Templates”, Computational Intelligence in Security for IS, Berlin, pp. 163–170. doi: https://doi.org/10.1007/978-3-642-04091-7_20.
4. Uludag, U., Pankanti, S., Prabhakar, S. and Jain, A. K. (2004), “Biometric cryptosystems: issues and challenges,” *Proceedings of the IEEE*, vol. 92, no. 6, pp. 948–960, Jun. 2004, doi: https://doi.org/10.1109/JPROC.2004.827372.Liu
5. Jin, Z., A. Teoh, B. J., Goi, B.-M. and Tay, Y.-H. (2016), “Biometric cryptosystems: A new biometric key binding and its implementation for fingerprint minutiae-based representation”, *Pattern Recognition*, vol. 56, pp. 50–62, Aug. 2016, doi: https://doi.org/10.1016/j.patcog.2016.02.024.
6. Fuller, B., Reyzin, L. and Smith, A. (2914), *When are Fuzzy Extractors Possible?* Available at: http://eprint.iacr.org/2014/961.
7. Lutsenko, M., Kuznetsov, A., Kiian, A., Smirnov, O. and Kuznetsova, T. (2021), “Biometric Cryptosystems: Overview, State-of-the-Art and Perspective Directions,” *Advances in Information and Communication Technology and Systems*, Cham, pp. 66–84. doi: https://doi.org/10.1007/978-3-030-58359-0_5.
8. Schneier, B. (1996), *Applied cryptography : protocols, algorithms, and source code in C*, Wiley, New York, available at: http://archive.org/details/appliedcryptogra00schn_328.
9. Menezes, A. J., van Oorschot, P. C., Vanstone, S. A., van Oorschot, P. C. and Vanstone, S. A. (2018), *Handbook of Applied Cryptography*. CRC Press, doi: https://doi.org/10.1201/9780429466335.
10. Klima, R. E. and Sigmon, N. P. (2018), *Cryptology : Classical and Modern*, Chapman and Hall/CRC, doi: https://doi.org/10.1201/9781315170664.
11. Rubinstein-Salzedo, S. (2018*), Cryptography*, Cham: Springer International Publishing, 2018. doi: https://doi.org/10.1007/978-3-319-94818-8.
12. Delfs, H. and Knebl, H. (2015), *Introduction to Cryptography*, Springer, Berlin, doi: https://doi.org/10.1007/978-3-662-47974-2.
13. Amin, R., Gaber, T., ElTaweel, G. and Hassanien, A. E. (2014), “Biometric and Traditional Mobile Authentication Techniques: Overviews and Open Issues”, *Bio-inspiring Cyber Security and Cloud Services: Trends and Innovations*, A. E. Hassanien, T.-H. Kim, J. Kacprzyk, and A. I. Awad, Eds. Springer, Berlin, pp. 423–446. doi: https://doi.org/10.1007/978-3-662-43616-5_16.
14. Jain, A. K., Ross, A. and Prabhakar, S. (2004), “An introduction to biometric recognition”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4–20, Jan. 2004, doi: https://doi.org/10.1109/TCSVT.2003.818349.
15. Clarke, N. L. and Furnell, S. M. (2007), “Advanced user authentication for mobile devices”, *Computers & Security*, vol. 26, no. 2, pp. 109–119, Mar. 2007, doi: https://doi.org/10.1016/j.cose.2006.08.008.
16. Geitgey, A. (2021), *ageitgey/face_recognition*, available at: https://github.com/ageitgey/face_recognition.
17. King, D. E. (2015), “Max-Margin Object Detection”, *arXiv:1502.00046 [cs]*, available at: http://arxiv.org/abs/1502.00046.

Received (Надійшла) 14.03.2023
Accepted for publication (Прийнята до друку) 24.05.2023

ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

Даценко Сергій Сергійович – магістрант кафедри комп'ютерної інженерії та програмування, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Serhii Datsenko – master’s degree student at Department of Computer Engineering and Programming, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine.
e-mail: sergdacenko@gmail.com; ORCID ID: https://orcid.org/0000-0001-9514-0433.

Кучук Георгій Анатолійович – доктор технічних наук, професор, професор кафедри комп'ютерної інженерії та програмування, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Heorhii Kuchuk – Doctor of Technical Sciences, Professor, Professor of the Department of Computer Engineering and Programming, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine.
e-mail: kuchuk56@ukr.net; ORCID ID: https://orcid.org/0000-0002-2862-438X.

Біометрична автентифікація, що використовує згорткові нейронні мережі

С. С. Даценко, Г. А. Кучук

Анотація. Актуальність. Криптографічні алгоритми та протоколи є важливими інструментами сучасної кібербезпеки. Вони використовуються в різних додатках, від простого програмного забезпечення для шифрування комп’ютерної інформації до складних інформаційних і телекомунікаційних систем, які реалізують різні електронні довірчі служби. Розробка повних біометричних криптографічних систем дозволить використовувати персональні біометричні дані як унікальний секретний параметр замість необхідності запам'ятовувати криптографічні ключі або використовувати додаткові пристрої аутентифікації. **Об’єкт дослідження** – процес генерації криптографічних ключів з біометричних зображень обличчя людини з реалізацією нечітких екстракторів. **Предмет дослідження** – засоби та методи побудови нейронної мережі з використанням сучасних технологій. **Метою даної статті** є дослідження нових методів генерації криптографічних ключів із біометричних зображень за допомогою згорткових нейронних мереж та гістограми орієнтованих градієнтів. **Результати дослідження.** Запропонована технологія дозволяє реалізувати новий криптографічний механізм – технологію генерації надійних криптографічних паролів з біометричних зображень для подальшого використання їх як атрибутів доступу до захищених систем, а також джерела ключів для існуючих криптографічних алгоритмів.

Ключові слова: біометричні криптографічні системи; криптографічні ключі; нечіткі екстрактори; згорткова нейронна мережа.