

DEVELOPMENT OF A REAL TIME HUMAN FACE RECOGNITION SOFTWARE SYSTEM

¹ASKAR BORANBAYEV, ²SEILKHAN BORANBAYEV, ²MUKHAMEDZHAN AMIRTAEV,
³MALIK BAIMUKHAMEDOV, ⁴ASKAR NURBEKOV

¹Nazarbayev University, Nur-Sultan, Kazakhstan

²L.N.Gumilyov Eurasian National University, Nur-Sultan, Kazakhstan

³Kostanay Socio-Technical University named after academician Z. Aldamzhar, Kostanay, Kazakhstan

⁴The University of Texas Rio Grande Valley, USA, Brownsville

E-mail: ¹aboranbayev@nu.edu.kz, ²sboranba@yandex.kz

ABSTRACT

In this study, a system for real-time face recognition was built using the Open Face tools of the Open CV library. The article describes the methodology for creating the system and the results of its testing. The Open CV library has various modules that perform many tasks. In this paper, Open CV modules were used for face recognition in images and face identification in real time. In addition, the HOG method was used to detect a person by the front of his face. After performing the HOG method, 128 face measurements were obtained using the image encoding method. A convolutional neural network was then used to identify people's faces using the SVM linear classifier algorithm.

Keywords: *System, Recognition, Method, Algorithm, Neural Network.*

1. INTRODUCTION

Today, most areas of science, technology and production are guided by the development of systems in which information has the character of an image. When processing this kind of information, a number of complex scientific, technical and technological problems arise. One of these tasks is image processing and recognition. Interest in image processing and recognition processes is relevant due to increasing practical needs: security systems, credit card verification, forensics, teleconferences, etc. Despite the fact that a person is good at identifying people's faces, the question arises of how to teach this to a computer, including how to decode and store digital images of faces.

Recognition of a human face in an image is the main and key in the tasks of recognizing emotions and automatically tracking people moving in the field of view of the camera. The problem of optimal search and identification of a human face, based on cybernetic vision systems, can be considered both in the light of the classical problem of perception, and in the light of new methods. The issue of localization and face recognition was studied at the early stages of computer vision. For over 30 years, many companies have been developing automatic systems for detecting and recognizing human faces: ASID

systems, FaceID; Imagis, Epic Solutions, Spillman, Trueface system, SMMA (Shoot Me My Account) authorization video recording system, etc. Face recognition tasks can be implemented using several approaches based on the following methods: statistical methods, graph theory, neural networks.

There are a number of reasons why the face recognition process is very difficult. For example, an object in the form of a person's face may be different due to different facial shapes and skin color, and other objects may partially overlap the face. The camera's ability to identify a face often depends on the quality of the video recording and the level of lighting. At the moment, some circumstances (such as camera quality, lighting, face angle, etc.) do not allow you to achieve the appropriate degree of recognition in images and video streams, and they also affect the quality of identification. In this regard, it is necessary to investigate what factors and parameters determine the quality of recognition and identification in real time and develop appropriate software systems.

In this study, a software system was developed using HOG, Viola-Jones and convolutional neural networks to improve real-time face recognition.

2. SYSTEM OF CLASSIFICATION AND RECOGNITION OF HUMAN FACE

2.1 Search for all faces in the photo: this is necessary in order to select the area of the image that is passed to further processing. A HOG (Histogram of Oriented Gradients) is used for this purpose[1].

According to this algorithm, the image is converted into black and white format, because color data is not needed to search for faces. The loop looks at each pixel and its neighboring pixels in order to find out how dark the current pixel is compared to the surrounding ones. Then an arrow is added to indicate which direction the image is getting darker. After performing this procedure for each individual pixel in the image, each pixel is replaced with an arrow. These arrows are called gradients, and they show the direction from light to dark pixels throughout the image. If you use dark and light images of the same person, the pixels will have different brightness values, but when you consider the direction of brightness changes, you get the same image regardless of the brightness of the original image. To save resources and get rid of redundant information, the image is divided into blocks of 16x16 pixels. Then replace this square in the image with arrows pointing in the same direction as the majority[2].

Eventually, the original image is transformed so that the basic structure of the face is clearly visible. To find a face in a HOG image, you need to find the part of the image that most closely resembles the well-known HOG drawing obtained from a variety of other faces during training. An example of a fragment that is highlighted in an image is shown in Figure 1.

2.2 Face position: to make it easier for the computer to work with faces turned, you must convert each image so that the eyes and lips are always in a specific place. To solve this problem, we use the face landmark estimation algorithm [4]. The basic idea is that there are 68 special points (called landmarks) that exist on each face – the upper part of the chin, the outer point of each eye, the inner point of each eyebrow, and so on. Then a machine learning algorithm is trained to find these 68 special points on any face. Figure 2 and 3 show the selected face landmarks.

When the eyes and mouth are defined in an image, you can rotate, zoom, and shift it so that the eyes and mouth are centered as best as possible. For

centering, only basic image transformations are used, such as rotation and scaling, which preserve parallel lines-affine transformations.

This method of image transformation is based on the following principle: for each point in the final image, a fixed set of points in the source image is taken and interpolated according to their relative positions. The affine transformation is the most General one-to-one mapping of a plane to a plane, which preserves straight lines and the ratio of lengths of segments lying on a single line. After this transformation, the face is centered in approximately the same position in the image, which makes the next step more accurate.

2.3 Face encoding: the face image obtained after the first steps must be "encoded" in a unique way, since comparing the resulting image with all the previous ones is an irrational approach when there is a large database of images. So we need a way to take a few basic measurements from each face, which we can compare with the closest known measurements and find the most similar face. In 1960, Woodrow Bledsoe proposed an algorithm that identifies obvious facial features. However, these measurements for the human brain (eye color, nose size, and so on) do not really make sense for a computer that looks at individual pixels in an image. Researchers have shown that the most accurate approach is to let the computer measure what it needs. Deep learning, determines which parts of the face to measure, better than people [6].

To solve this problem, a convolutional deep learning neural network is created that generates 128 dimensions (face map) for each person (Figure 4). The idea of converting images into a list of computer-generated numbers is extremely important for machine learning.

The algorithm of image analysis:

3 images are analyzed to train the network:

1. Training image of a famous person's face
2. Another photo of the same famous person
3. An image of a completely different person

The algorithm then looks at the measurements it makes for each of these three images. Then it tweaks the neural network a little to make sure that the dimensions created for images #1 and #2 are more similar, and the dimensions for #2 and #3 are less similar. A schematic representation of the algorithm is shown in Figure 5.

After repeating this step m times for n images of different people, the neural network is able to reliably create 128 characteristics for each person. Any 10-15 different images of the same person may well give the same characteristics.

The algorithm allows you to search the database for an image that has characteristics that are closest to the characteristics of the image you are looking for [8].

By training a network with different images of the same person, the algorithm creates approximately the same dimensions. The resulting 128 dimensions are called a face map. The idea of converting an array of raw data, such as an image, into a list of computer-generated numbers is extremely important for machine learning.

2.4 Search name by face map: the SVM linear classifier algorithm (Support Vector Method) is used to search for a person's name on a face map. This algorithm is based on the concept of the solution plane. The plane divides the received data with different classes. One of the main advantages of SVM is the high speed of learning the algorithm, and, consequently, the ability to use a fairly large amount of source data for training. Using this method solved the problem of binary classification, in this example determined the identity photographs to the existing sample. To do this, you need to train a classifier that takes measurements from the image being checked and shows who this person is most similar to from the entire training sample of the system.

Other classifiers include the RVM - Relevance Vector Machine method. In contrast to SVM, this method gives the probabilities with which the face image belongs to a given person. i.e., if SVM says "the image belongs to person 1", then RVM will say "the image belongs to person 1 with probability p and class person 2 with probability $1-p$ " [9]. In addition, the reference vector method is unstable with respect to noise in the source data. If the training sample contains noise emissions, they are significantly taken into account when constructing the dividing hyperplane. The method of relevant vectors does not have this disadvantage.

Despite the existence of various algorithms, it is possible to distinguish a General scheme of the face recognition process as shown in Figure 6.

Before you start the face recognition process, you must complete the following steps:

- localize faces in an image
- to align the image of the face (geometrical and luminance)
- detect signs
- face recognition itself is a comparison of the found features with the standards previously included in the database.

3. CONVOLUTIONAL NEURAL NETWORK FOR SOLVING THE RECOGNITION PROBLEM

The pre-trained model is a convolutional neural network (CNN), a type of neural network that builds state-of-the-art models for computer vision. The CNN which is used in this article itself was trained by "Davis King" on a dataset approximately of 1.2 million images and evaluated on the ImageNet classification dataset that consist of 1000 classes. Architecture of this pre-trained model is based on ResNet-34, with 34 layers (Figure 7).

The convolutional layers of ResNet (Figure 7 - the third column) is based on the plain network (Figure 8 - the second column). But the difference with Plain Network is shortcut connections, which turns the network into a residual version of it. Shortcut connections skip one or more layers and perform ID mapping. Their outputs are added to the outputs of the layers being laid (Figure 8).

Here is simple way of using ResNet-34 for face recognition:

- Firstly, make sure that you have installed Python, OpenFace, dlib.
- Then, execute this command: `pip3 install face_recognition`
- After that you have to create 2 folders, "pictures_of_people_i_know" and "unknown_pictures". First folder should contain people that you already know, second one should contain people that you want to identify.
- The next step is running the command `face_recognition` that passes through both folders, and tells you who is in each image:


```
face_recognition ./pictures_of_people_i_know/
./unknown_pictures/ [11]
```

The results will look like this:
`./unknown_pictures/obama.jpg,Barack_Obama.jpg` - "obama.jpg" and "Barack_Obama.jpg" are the same person

/face_recognition_test/unknown_pictures/unknown.jpg, unknown_person - did not find any person [11]

4. REAL TIME HUMAN FACE RECOGNITION SYSTEM

The software system uses the HOG method to solve the problem of detecting faces in images, and convolutional neural networks are used for the recognition problem.

The following methods were used for conducting the work:

- Computer: MacBook Pro 13 2015 laptop, Intel Core i5-5257U CPU@2.7GHz, 8 Gb RAM; Intel Iris Graphics 6100 1536 MB graphics card

- Operating system: Mac OS 10.13.6, 64-bit

- Python 3.7.6 programming language; libraries for python-dev development; NumPy library for scientific computing; dlib library, OpenFace[12]; Git-distributed version control system.

Table 1: - Comparison of ResNet models [13]

Model	Training accuracy	Validation accuracy	Loss	Validation Loss	Time
ResNet18	0.83	0.87	1.0436	1.006	2701s
ResNet34	0.8651	0.8519	1.5983	1.7510	4800s
ResNet50	0.8662	0.8095	4.3967	4.5914	5580s
ResNet101	0.8594	0.7884	8.4274	8.7475	6112s
ResNet152	0.8798	0.8836	11.943	12.05	9248s

To evaluate this model, we need to consider other ResNet models with different layers. The ResNet CNN model could contain 152 layers. In this research, TensorFlow ResNet Library is used for its simple deployment and for obtaining accuracy. An ultimate training accuracy of 0.8651 and validation of 0.8519 was discovered in the model. Experimentally determined ultimate measures of loss was 1.5983 whereas validation loss was 1.006. Table 1 provides a summary results of the analyzed ResNet models. Next, Figure 9 and Figure 10 summarize time accuracy and compare training and testing accuracy of the analyzed models.

The task of the developed software system for real-time face recognition is the detection, identification and tracking of faces that appear on the video. For example, a software system can be used to search for a specific category of people. To do this, first the relevant information about these people is entered into the database. The system allows you to detect a person from a video stream from a camera, make a search in the database and identify him, if he is in the database. When developing a software system, special methods and technologies

were used to optimize it and ensure reliability and safety, described in works [14-51].

The real-time facial recognition system was created using the OpenCV 3 library and the Python 3.7 programming language, as well as the Face Recognition library. Training a facial recognition system using an image of a person's face is shown in Figure 11.

Figure 11 shows the code of the face recognition system for training images, as well as folders divided into 3 categories, such as 'groups' (photos of several people), 'known' (photos trained by the face recognition system), 'unknown' (people unknown to the system). In Figure 11, you can see functions such as `face_recognition.load_file_image`, which loads images, and `face_recognition.face_locations`, a function that can be used to find faces in an image. After that, the found faces are saved in such lists as `known_face_encodings`, which stores the universal "encoding" of facial features (128 dimensions-face map) and `known_face_names` the names of these people corresponding to `known_face_encodings`. The `face_recognition.compare_faces` function is used, which compares the faces in the image / video stream with the images in the database.

The face recognition system, using a webcam, takes real-time screenshots of the faces that need to be recognized. Screenshots of faces are taken using the Directed Gradient Histogram method. The word "screenshots" refers to the location of the faces in the camera's view in each frame. Using the location of faces, you can find the face itself in the image / video stream. It is stored in an object called list. Video cameras shoot an average of 30 frames per second. The system searches for faces in each of these frames, and an example of the code for finding faces is shown in Figure 12.

Figure 12 shows the face search code, where `face_locations` is a list that stores the location of the faces of each frame, `face_encodings` contains a universal "encoding" of facial features (128 dimensions-face map), with this encoding you can compare the encoding of images that are in the database.

Tests were carried out to verify the system for identifying people in the video stream, the results of which showed a high quality of recognition of several faces in the frame at the same time.

5. CONCLUSION

When developing a human face recognition system in real time, the methods of HOG, Viola-Jones, convolutional neural networks, Open Face tools of the Open CV library were used.

A study was carried out on what factors and parameters determine the quality of recognition and identification in real time, and a corresponding software system was developed.

The results of this study showed that the created system allows identifying the faces of several people directly in real time. System testing has shown that poor lighting can negatively affect recognition and identification, while good lighting has a positive effect. The angle of view of the face and the distance between the person being tested and the camera can also negatively affect the recognition and identification results. When using high-quality video cameras, the quality of identification and identification is significantly improved.

Subsequent research in this area can be aimed at improving the quality of recognition and identification in real time. For further study of the dependence of the quality of recognition and identification on weather conditions, the resolving properties of the camera, etc.

REFERENCES

- [1] Navneet Dalal, Bill Triggs, Histograms of Oriented Gradients for Human Detection [Electronic source] // URL: <http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf>.
- [2] "Extract HOG Features". [Electronic source] URL: <https://www.mathworks.com/help/vision/ref/extracthogfeatures.html>
- [3] M. Yang, N. Ahuja and D. Kriegman. Face recognition using kernel eigenfaces. Image Processing: IEEE Transactions - 2000. - Vol.1. pp. 37- 40
- [4] V. Kazemi and S. Josephine. One millisecond face alignment with an ensemble of regression trees. In CVPR, 2014. [Electronic source] // URL: <http://www.csc.kth.se/~vahidk/papers/Kazemi-CVPR14.pdf>.
- [5] Facial recognition. [Electronic source] URL: <https://ai.nal.vn/facial-recognition/>
- [6] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In Proc. CVPR, 2015. [Electronic source] // URL: https://www.cv-foundation.org/openaccess/content_cvpr_2015/app/1A_089.pdf.
- [7] Neha Rudraraju, Kotoju Rajitha, K. Shirisha, Constructing Networked, Intelligent and Adaptable Buildings using Edge Computing [Electronic source] // URL: http://ijrar.com/upload_issue/ijrar_issue_20542_753.pdf
- [8] Dalal N., Triggs B. Histograms of Oriented Gradients for Human Detection // Proc. of the IEEE Conference Computer Vision and Pattern Recognition. 2005. pp. 886–893.
- [9] Christopher M. Bishop F.R.Eng. Pattern Recognition and Machine Learning. [Electronic source] // URL: <https://goo.gl/WLqpHN>.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition [Electronic source] // URL: <https://arxiv.org/abs/1512.03385>
- [11] Face recognition [Electronic source] // URL: https://github.com/ageitgey/face_recognition
- [12] Howse J. OpenCV Computer vision with Python. – Packt Publishing Ltd., UK. 2013.
- [13] Riaz Ullah Khan, Xiaosong Zhang, Rajesh Kumar. Analysis of ResNet and GoogleNet models for malware detection [Electronic source] // URL: https://www.researchgate.net/publication/327271897_Analysis_of_ResNet_and_GoogleNet_models_for_malware_detection
- [14] Boranbayev, A., Boranbayev, S., Nurusheva, A., Yersakhanov, K. Development of a software system to ensure the reliability and fault tolerance in information systems. Journal of Engineering and Applied Sciences. – 2018, 13 (23), pp. 10080-10085.
- [15] Boranbayev, S., Goranin, N., Nurusheva, A. The methods and technologies of reliability and security of information systems and information and communication infrastructures. Journal of Theoretical and Applied Information Technology. -2018, 96 (18), pp. 6172-6188.
- [16] Boranbayev, A., Boranbayev, S., Nurusheva, A. Development of a software system to ensure the reliability and fault tolerance in information systems based on expert estimates. -2018, Advances in Intelligent Systems and Computing, Vol. 869, pp. 924-935.
- [17] Boranbayev, A., Boranbayev, S., Yersakhanov, K., Nurusheva, A., Taberkhan, R. Methods of Ensuring the Reliability and Fault Tolerance of Information Systems. Advances in Intelligent

- Systems and Computing. -2018, Vol. 738, pp. 729-730.
- [18] Boranbayev, S., Altayev, S., Boranbayev, A. Applying the method of diverse redundancy in cloud based systems for increasing reliability. The 12th International Conference on Information Technology: New Generations (ITNG 2015). April 13-15, 2015, Las Vegas, Nevada, USA, pp. 796-799.
- [19] Turskis, Z., Goranin, N., Nurusheva, A., Boranbayev, S. A fuzzy WASPAS-based approach to determine critical information infrastructures of EU sustainable development. -2019, Sustainability (Switzerland), Volume 11, Issue 2, 424.
- [20] Turskis, Z., Goranin, N., Nurusheva, A., Boranbayev, S. Information security risk assessment in critical infrastructure: A hybrid MCDM approach. -2019, Informatica (Netherlands), Volume 30, Issue 1, pp. 187-211.
- [21] Boranbayev, A. S., Boranbayev, S. N., Nurusheva, A. M., Yersakhanov, K. B., Seitkulov, Y. N. Development of web application for detection and mitigation of risks of information and automated systems. Eurasian Journal of Mathematical and Computer Applications. -2019, Volume 7, Issue 1, pp. 4-22.
- [22] Boranbayev, A.S., Boranbayev, S.N., Nurusheva, A.M., Seitkulov, Y.N., Sissenov, N.M. A method to determine the level of the information system fault-tolerance. Eurasian Journal of Mathematical and Computer Applications. -2019, Volume 7, Issue 3, pp. 13-32.
- [23] Boranbayev Askar, Boranbayev Seilkhan, Nurbekov Askar, Taberkhan Roman. The Development of a Software System for Solving the Problem of Data Classification and Data Processing. 16th International Conference on Information Technology - New Generations (ITNG 2019). -2019, Volume 800, pp. 621-623.
- [24] Boranbayev Askar, Boranbayev Seilkhan, Nurusheva Assel, Yersakhanov Kuanysh, Seitkulov Yerzhan. A software system for risk management of information systems. Proceedings of the 2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT 2018), 17-19 Oct. 2018, Almaty, Kazakhstan, pp. 284-289.
- [25] Boranbayev Seilkhan, Boranbayev Askar, Altayev Sanzhar, Nurbekov Askar. Mathematical model for optimal designing of reliable information systems. Proceedings of the 2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT 2014), Astana, Kazakhstan, October 15-17, 2014, pp. 123-127.
- [26] Boranbayev Seilkhan, Altayev Sanzhar, Boranbayev Askar, Seitkulov Yerzhan. Application of diversity method for reliability of cloud computing. Proceedings of the 2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT 2014), Astana, Kazakhstan, October 15-17, 2014, pp.244-248.
- [27] Boranbayev A., Boranbayev S., Nurusheva A. Analyzing methods of recognition, classification and development of a software system. Advances in Intelligent Systems and Computing. -2018, Vol. 869, pp. 690-702.
- [28] Boranbayev, A.S., Boranbayev, S.N. Development and optimization of information systems for health insurance billing. ITNG2010 - 7th International Conference on Information Technology: New Generations. - 2010, pp. 1282-1284.
- [29] Akhmetova, Z., Zhuzbayev, S., Boranbayev, S., Sarsenov, B. Development of the system with component for the numerical calculation and visualization of non-stationary waves propagation in solids. Frontiers in Artificial Intelligence and Applications. -2016, Vol. 293, pp. 353-359.
- [30] Boranbayev, S.N., Nurbekov, A.B. Development of the methods and technologies for the information system designing and implementation. Journal of Theoretical and Applied Information Technology. -2015, 82 (2), pp. 212-220.
- [31] Boranbayev, A., Shuitenov, G., Boranbayev, S. The method of data analysis from social networks using apache hadoop. Advances in Intelligent Systems and Computing. -2018, Vol. 558, pp. 281-288.
- [32] Boranbayev, S., Nurkas, A., Tulebayev, Y., Tashtai, B. Method of Processing Big Data. Advances in Intelligent Systems and Computing. -2018, Vol. 738, pp. 757-758.
- [33] Boranbayev, A., Boranbayev, S., Nurbekov, A. Estimation of the Degree of Reliability and Safety of Software Systems. Advances in Intelligent Systems and Computing. -2020, Vol. 1129, AISC, pp. 743-755.
- [34] Boranbayev, A., Boranbayev, S., Nurbekov, A. Development of the Technique for the Identification, Assessment and Neutralization of Risks in Information Systems. Advances in

- Intelligent Systems and Computing. -2020, Vol. 1129, AISC, pp. 733-742.
- [35] Boranbayev, A., Boranbayev, S., Nurusheva, A., Seitkulov, Y., Nurbekov, A. Multi Criteria Method for Determining the Failure Resistance of Information System Components. *Advances in Intelligent Systems and Computing*. -2020, Vol. 1070, pp. 324-337.
- [36] Boranbayev, A., Boranbayev, S., Nurbekov, A., Taberkhan, R. The Software System for Solving the Problem of Recognition and Classification. *Advances in Intelligent Systems and Computing*. -2019, Vol. 997, pp. 1063-1074.
- [37] Akhmetova, Z., Boranbayev, S., Zhuzbayev, S. The visual representation of numerical solution for a non-stationary deformation in a solid body. *Advances in Intelligent Systems and Computing*. -2016, Vol. 448, pp. 473-482.
- [38] Boranbayev, A., Boranbayev, S., Nurbekov, A. Evaluating and Applying Risk Remission Strategy Approaches to Prevent Prospective Failures in Information Systems. *Advances in Intelligent Systems and Computing*. -2020, Vol. 1134, pp. 647-651.
- [39] Boranbayev, A., Boranbayev, S., Nurbekov, A. A Proposed Software Developed for Identifying and Reducing Risks at Early Stages of Implementation to Improve the Dependability of Information Systems. *Advances in Intelligent Systems and Computing*. -2020, Vol. 1134, pp. 163-168.
- [40] Boranbayev, A., Boranbayev, S., Nurbekov, A. Java Based Application Development for Facial Identification Using OpenCV Library. *Advances in Intelligent Systems and Computing*, 2021, Vol. 1251, pp. 77-85.
- [41] Boranbayev, A., Boranbayev, S., Nurbekov, A. Measures to ensure the reliability of the functioning of information systems in respect to state and critically important information systems. *Advances in Intelligent Systems and Computing*, 2021, Vol. 1252, pp. 139-152.
- [42] Boranbayev, A., Boranbayev, S., Nurbekov, A. Development of a Hardware-Software System for the Assembled Helicopter-Type UAV Prototype by Applying Optimal Classification and Pattern Recognition Methods. *Advances in Intelligent Systems and Computing*, 2020, Vol.1229, pp. 380-394.
- [43] Boranbayev, A., Shuitenov, G., Boranbayev, S. The Method of Analysis of Data from Social Networks Using Rapidminer. *Advances in Intelligent Systems and Computing*, 2020, Vol. 1229, pp. 667-673.
- [44] Boranbayev Askar, Boranbayev Seilkhan, Yatsenko Yuri, Tulebayev Yersultan. Methods and software for simulation of optimal renewal of capital assets // *Journal of Theoretical and Applied Information Technology (JATIT)*. - 2020, Vol. 98 (21), pp. 3545-3558.
- [45] Boranbayev, A., Boranbayev, S., Nurusheva, A., Yersakhanov, K. The Modern State and the Further Development Prospects of Information Security in the Republic of Kazakhstan. *Advances in Intelligent Systems and Computing*, 2018, Vol. 738, pp. 33-38.
- [46] Akhmetova, Z., Zhuzbaev, S., Boranbayev, S. The method and software for the solution of dynamic waves propagation problem in elastic medium. *Acta Physica Polonica A*, 2016, 130(1), pp. 352-354.
- [47] Hritonenko, N., Yatsenko, Y., Boranbayev, S. Environmentally sustainable industrial modernization and resource consumption: Is the Hotelling's rule too steep? *Applied Mathematical Modelling*, 2015, 39(15), pp. 4365-4377.
- [48] Yatsenko, Y., Hritonenko, N., Boranbayev, S. Non-equal-life asset replacement under evolving technology: A multi-cycle approach. *Engineering Economist*, 2020, 65(4), pp. 339–362.
- [49] Boranbayev, S.N., Nurbekov, A.B. Construction of an optimal mathematical model of functioning of the manufacturing industry of the republic of Kazakhstan. *Journal of Theoretical and Applied Information Technology*, 2015, Vol. 80(1), pp. 61-74.
- [50] Boranbayev, A., Boranbayev, S., Muratov, T., Nurbekov, A. Modeling Dependence Between Air Transportation and Economic Development of Countries. *Advances in Intelligent Systems and Computing*, 2021, Vol. 1289, pp. 601-613.
- [51] Boranbayev, A., Boranbayev, S., Seitkulov, Y., Nurbekov, A. Proposing Recommendations for Improving the Reliability and Security of Information Systems in Governmental Organizations in the Republic of Kazakhstan. *Advances in Intelligent Systems and Computing*, 2021, Vol. 1290, pp. 854-868.

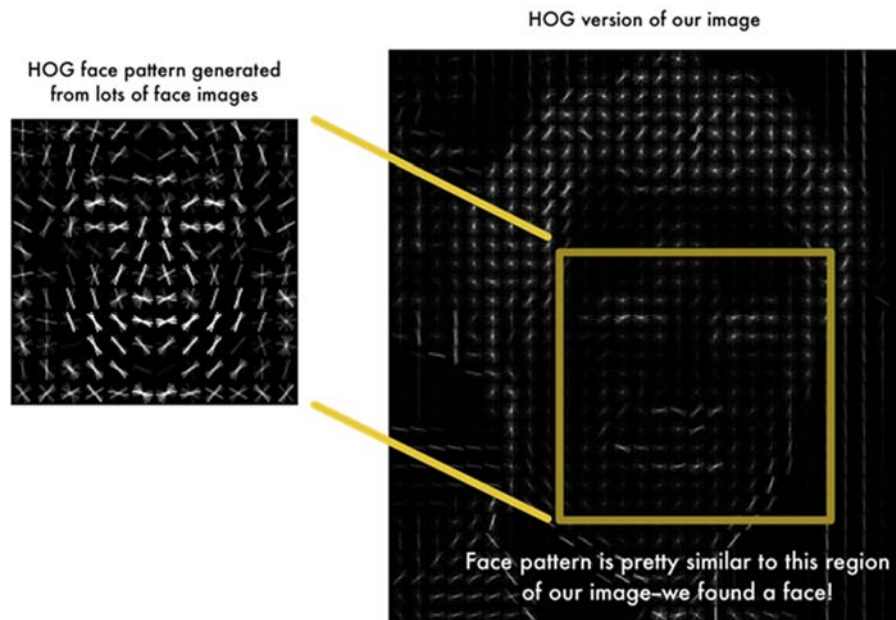


Figure 1. Search For A Face In A Photo [3]

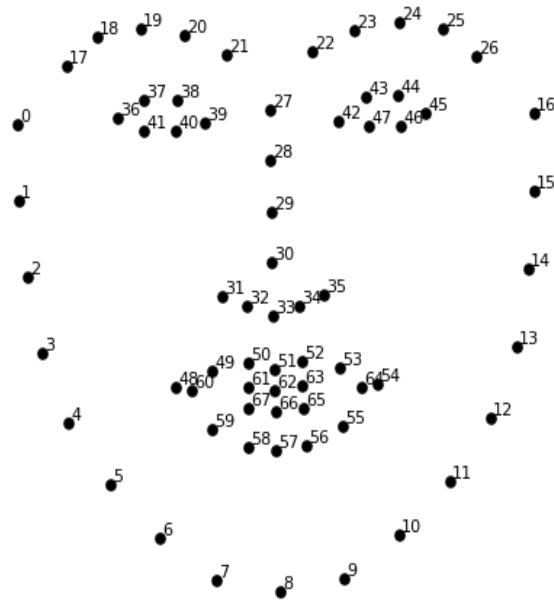


Figure 2. 68 Face Landmarks [3]

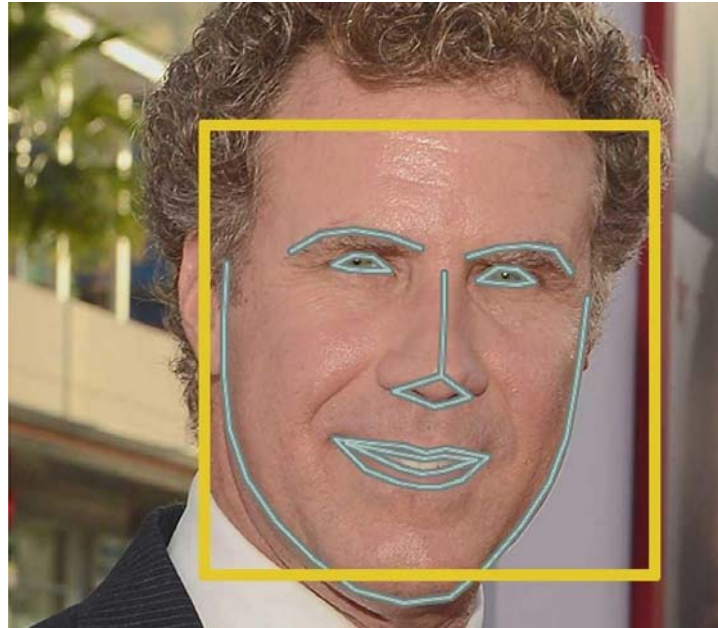


Figure 3. The Identification Of The Landmarks Of The Face [5]

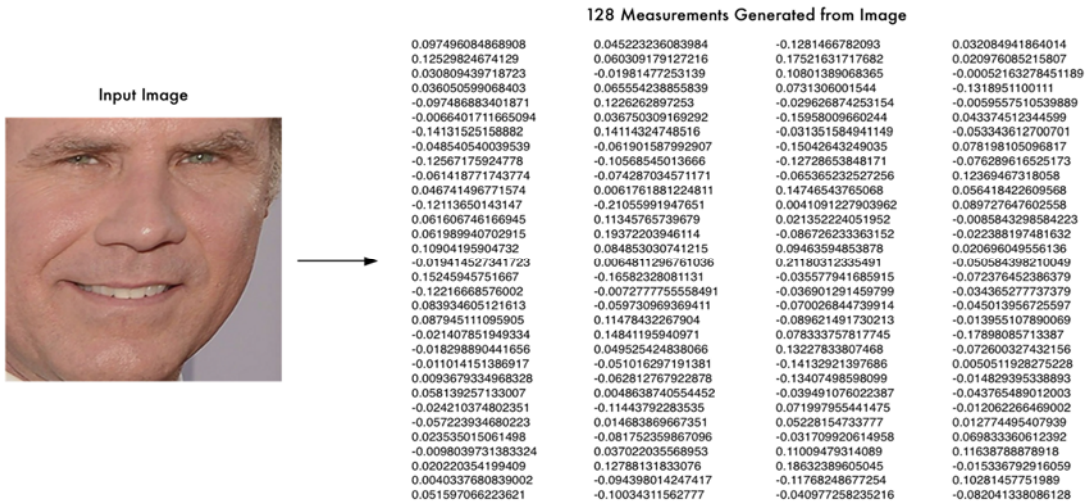


Figure 4. Measurement Results For The Test Image [7]

A single 'triplet' training step:

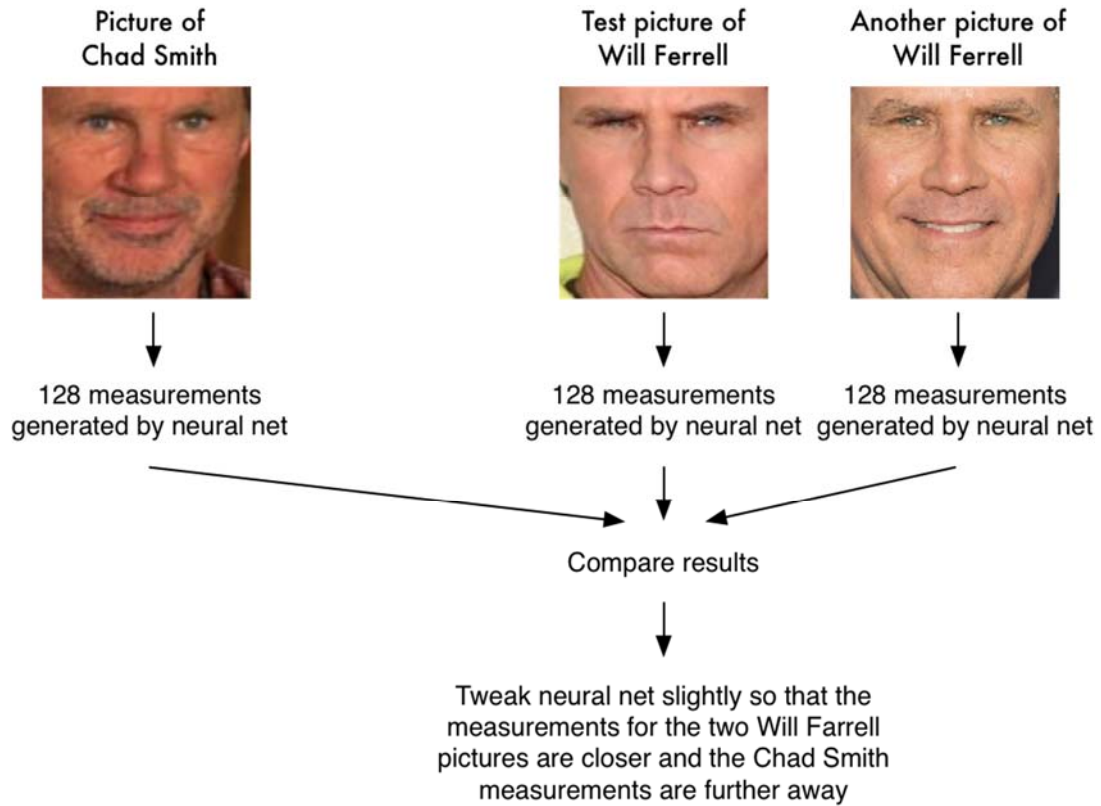


Figure 5. Demonstration Of The Algorithm [7]

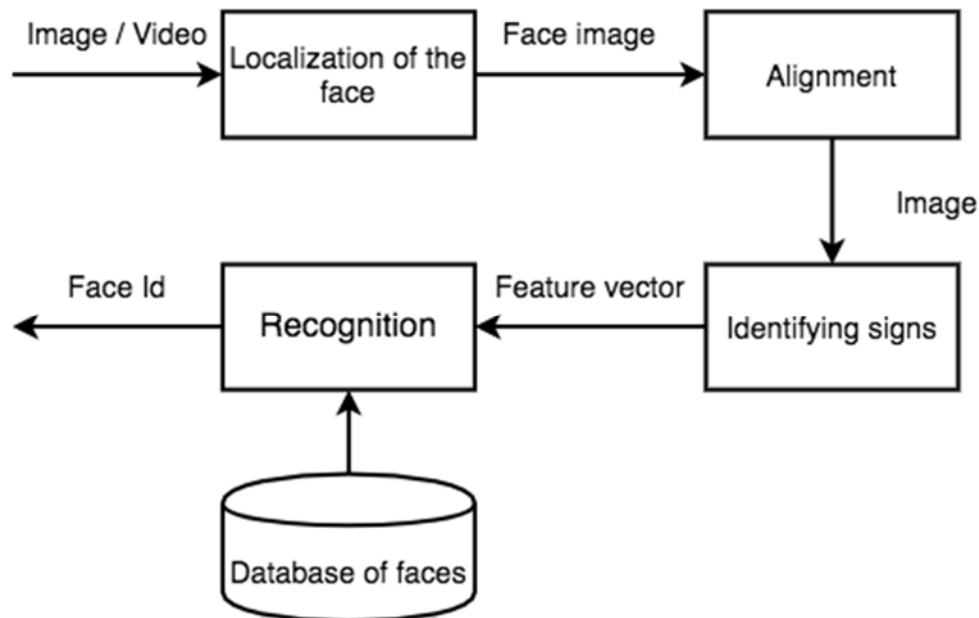


Figure 6. Diagram Of The Face Recognition Process

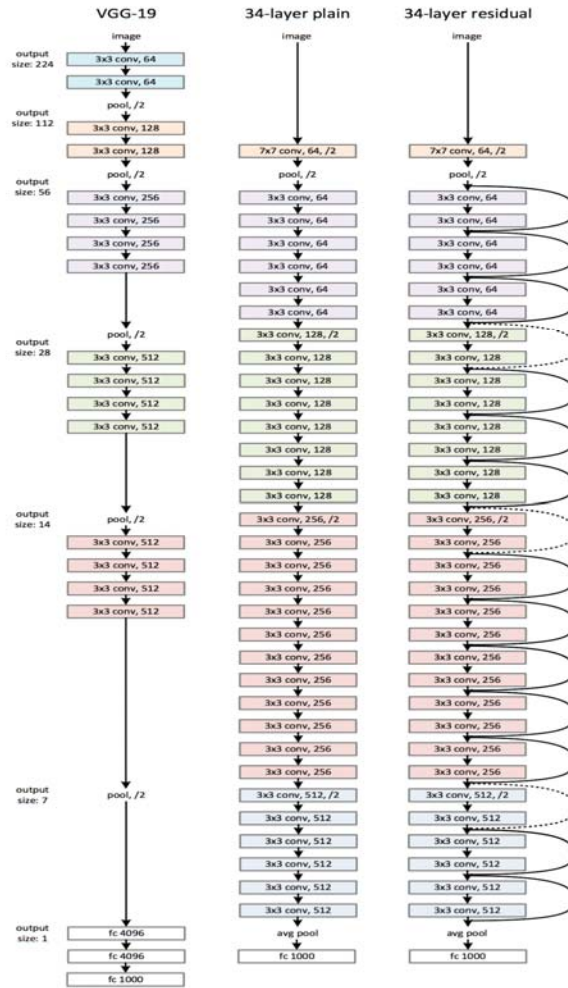


Figure 7. Architecture Of Resnet [10]

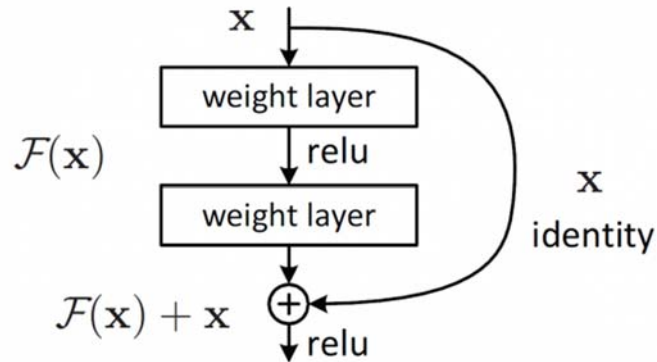


Figure 8. Shortcut Connections [10]

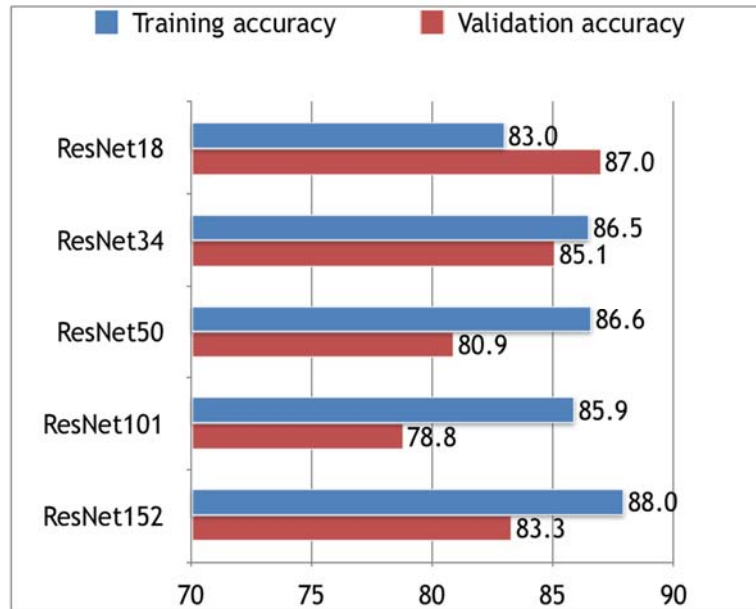


Figure 9. Training And Testing Accuracy [13]

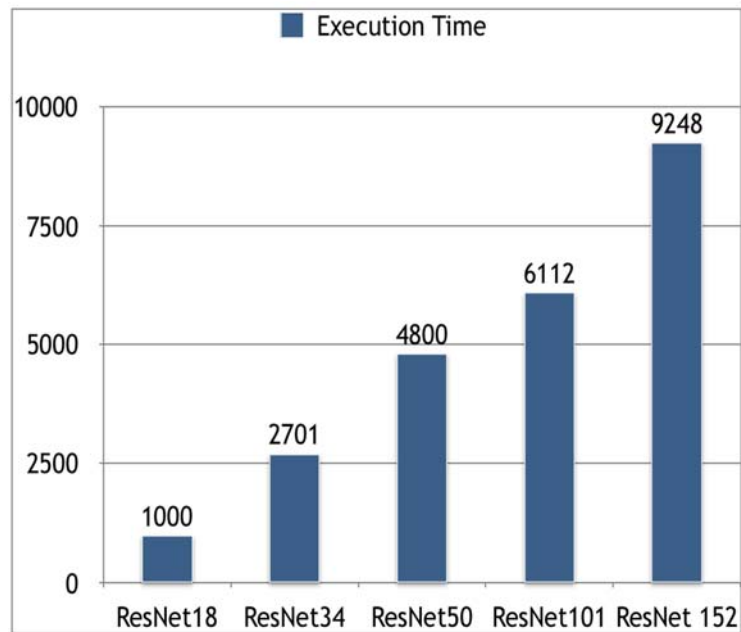
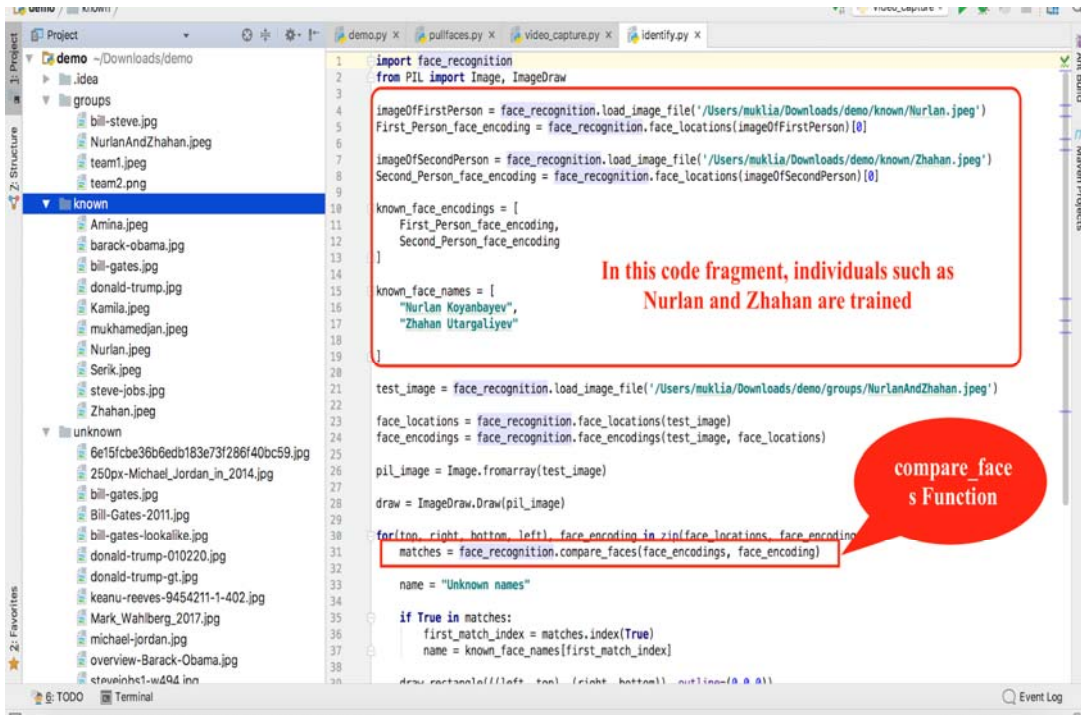


Figure 10. Execution Time Of Models [13]



```

1 import face_recognition
2 from PIL import Image, ImageDraw
3
4 imageOfFirstPerson = face_recognition.load_image_file('/Users/mukLia/Downloads/demo/known/Nurlan.jpeg')
5 First_Person_face_encoding = face_recognition.face_locations(imageOfFirstPerson)[0]
6
7 imageOfSecondPerson = face_recognition.load_image_file('/Users/mukLia/Downloads/demo/known/Zhahan.jpeg')
8 Second_Person_face_encoding = face_recognition.face_locations(imageOfSecondPerson)[0]
9
10 known_face_encodings = [
11     First_Person_face_encoding,
12     Second_Person_face_encoding
13 ]
14
15 known_face_names = [
16     "Nurlan Koyanbayev",
17     "Zhahan Utargaliyev"
18 ]
19
20 test_image = face_recognition.load_image_file('/Users/mukLia/Downloads/demo/groups/NurlanAndZhahan.jpeg')
21
22 face_locations = face_recognition.face_locations(test_image)
23 face_encodings = face_recognition.face_encodings(test_image, face_locations)
24
25 pil_image = Image.fromarray(test_image)
26
27 draw = ImageDraw.Draw(pil_image)
28
29 for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
30     matches = face_recognition.compare_faces(face_encodings, face_encoding)
31
32     name = "Unknown names"
33
34     if True in matches:
35         first_match_index = matches.index(True)
36         name = known_face_names[first_match_index]
37
38     draw.rectangle(((left, top), (right, bottom)), outline=(0, 0, 0))
    
```

In this code fragment, individuals such as Nurlan and Zhahan are trained

compare face s Function

Figure 11. Image Training

```

66 # Initialize some variables
67 face_locations = []
68 face_encodings = []
69 face_names = []
70 process_this_frame = True
71
72 while True:
73     # Grab a single frame of video
74     ret, frame = video_capture.read()
75
76     # Resize frame of video to 1/4 size for faster face recognition processing
77     small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
78
79     # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
80     rgb_small_frame = small_frame[:, :, :-1]
81
82     # Only process every other frame of video to save time
83     if process_this_frame:
84         # Find all the faces and face encodings in the current frame of video
85         face_locations = face_recognition.face_locations(rgb_small_frame)
86         face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
87
88         face_names = []
89         for face_encoding in face_encodings:
90             # See if the face is a match for the known face(s)
91             matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
92             name = "Unknown"
93
94             face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
95             best_match_index = np.argmin(face_distances)
96             if matches[best_match_index]:
97                 name = known_face_names[best_match_index]
98
99             face_names.append(name)
100
101     process_this_frame = not process_this_frame
102
    
```

Process of Finding faces

Figure 12. The Process Of Finding The Faces In Each Frame