Software Tool for Recognition of Human Face in Video Stream

Svitlana Popereshnyak[1] , Rodion Skoryk [1], Dmytro Kuptsov[2] and Roman Kravchenko[2]

[1] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37, Prospect Beresteiskyi, Kyiv, 03056, Ukraine*
[2] *Institute of Software Systems of NAS of Ukraine, Academician Glushkov Avenue, 40, Kyiv, 03187, Ukraine*

Abstract

In the work, an analysis of detection methods and faces in the video stream and their effectiveness in real time was carried out. Modern algorithms and pre-trained models have been found to be able to recognize faces with high accuracy, but their significant drawback is, in particular, vulnerability to attacks using fake faces. Therefore, the work also analyzed approaches to detecting living faces and the possibility of their implementation in the system. Using an object-oriented approach, a tool for face capture, receiving a video stream from various sources, detecting unknown and previously captured faces in a video stream, and recognizing live faces was designed and developed. The system has been adapted to work in real time using the GPU. The work improved the architecture of a convolutional neural network for recognizing living faces with the creation of a dataset from a combination of own footage and open datasets. Also, a user interface for the face recognition system was developed. The work improved identification procedures and simplified detection of persons on video for employees of the security department of enterprises by implementing liveness detection face recognition methods. As a result of the research, a system was designed, which is intended for detection, recognition and detection of living faces in a video stream. After analyzing the known successful software products, niches that need a new solution were identified. Based on them, functional and non-functional requirements were developed. The process of recognizing faces in the video stream has been modified by implementing our own Liveness Detection model.

Keywords

Face detection, face recognition, live face detection, deep learning, video streaming, Internet of Things, state border, streaming information, software complexes, observability and controllability

1. Introduction

Facial recognition is a biometric security system, along with voice recognition, fingerprint recognition, and retina recognition. The biological process of recognizing faces in humans is very interesting, because a person can still recognize someone even if they have grown or undergone significant changes in appearance. With the help of computer technology, this is also possible, provided the algorithm is of sufficient quality. A problem that often occurs in facial recognition systems is that it is easy to fool the system with fake faces or images of human faces, such as photos, videos, masks, and statues, which is called a spoofing attack. This is possible because the basic principle of face recognition is to remember unique information about the face, encode it and compare it with previously encoded results. Using an image of a human face for identification or something that cannot be detected without LD: attackers can use a photo of a person already registered in the system to bypass the facial recognition system and gain personal gain. To make the face recognition system more secure, we need an algorithm that can determine whether the user

14th International Scientific and Practical Conference from Programming UkrPROG'2024, May 14-15, 2024, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
spopereshnyak@gmail.com (S. Popereshnyak); pipom82015@gmail.com (R. Skoryk); dmitry.v.kuptsov@gmail.com (D. Kuptsov); krav.rom.vik@gmail.com (R. Kravchenko)
0000-0002-0531-9809 (S. Popereshnyak); 0009-0000-9547-4038 (R. Skoryk); 0009-0009-9958-6809 (D. Kuptsov); 0009-0005-8044-4414 (R. Kravchenko)
© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-3806/S_57_Popereshnyak_Skoryk_Kuptsov_Kravchenko.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

is real or fake, which is called liveness detection. Unlike regular face recognition without this feature, which just takes data from the camera without checking if it's a real person, this solves a potential drawback: regular face recognition treats everything in front of it as a real human face, even if it's just an image or an object. This is where live human detection comes in to fix this [1].

LD is used in sectors where high security is critical: in financial institutions, at checkpoints, in access control systems and mobile applications.

Among the main challenges of these technologies is ensuring the accuracy and speed of processing, as well as taking into account ethical issues related to data confidentiality and privacy.

The relevance of the work is due to the growing popularity of face recognition as a biometric approach and as one of the most effective and convenient to use. This work is relevant because modern algorithms and pre-trained models allow high-accuracy face recognition, but their significant drawback is, in particular, vulnerability to attacks using fake faces. However, there is no open solution for live face recognition, so it is urgent to develop your own solution to meet the security needs.

2. Overview of industries requiring systems for recognizing individuals

Technological progress leads to a significant increase in the number of cybercrimes. Almost everyone in the world has digital accounts that contain sensitive personal information, often protected only by a simple password. Hence, security systems play a key role in ensuring privacy. It is necessary to have a reliable system that can distinguish between people and apply appropriate rights depending on their identification. Unlike standard authentication methods such as passwords, email verification, or the use of fingerprints, biometric facial recognition uses unique mathematical and dynamic patterns, making such systems one of the most secure and effective.

One of the industries that needs a system to recognize people and grant them the appropriate rights is the Internet of Things (IoT) [2]. The relevance of facial recognition in the Internet of Things (IoT) is determined by several factors:

1. Growth of IoT: The proliferation of IoT devices in all areas of life, from home devices to industrial systems, creates the need for security, user identification and process automation. Facial recognition can become a key technology for these tasks.
2. Increasing the level of security: In connection with the increase in the number of cybercrimes and security incidents, it is necessary to develop more effective and reliable methods of authentication and access control. Facial recognition can provide a high level of security because the face is a unique biometric identifier.
3. Convenience for users: The use of facial recognition can make interaction with IoT devices more convenient for users. They can automatically recognize users and adapt to their preferences and needs.
4. Improve data-driven decisions: Collecting and analyzing data about user behavior in real time can help optimize processes, improve customer interactions, and create personalized services. Facial recognition helps collect this data directly and efficiently.

Therefore, facial recognition in IoT is important both to ensure security and convenience for users, and to improve the functionality and efficiency of IoT systems as a whole.

The system for recognizing human faces in a video stream can be a key component in optimizing the quality control of streaming information about crossing the state border using artificial intelligence methods.

The integration of the face recognition system allows to automate the process of detection and identification of persons crossing the border, which makes control more effective and faster. The use of artificial intelligence methods allows to improve face recognition algorithms and increase the accuracy of detection.

The combination of these technologies makes it possible to create a system capable of detecting and identifying persons in a video stream in real time, thus optimizing the quality control of streaming information regarding the crossing of the state border. This approach makes it possible to ensure high efficiency and reliability of control at the state border, reducing the possibility of the passage of unwanted persons or objects across the border.

The integration of a facial recognition system can significantly increase the efficiency of the information systems used by the State Border Service of Ukraine (SBSU), allowing to automatically detect and track persons crossing the border or performing other actions that require attention.

In addition, the use of a facial recognition system can assist in responding to potential threats and identifying individuals who are wanted or have criminal intent.

The observability and controllability of software complexes for recognizing people's faces in a video stream is a mandatory characteristic of such systems and is ensured at the stage of choosing and implementing architectural solutions for creating appropriate software systems. This approach to the construction of software complexes ensures an increased level of security and control at the border.

The purpose of the work is to improve the identification procedure and simplify the identification of people on video for employees of the security department of enterprises by implementing liveness detection face recognition methods.

Face recognition, especially in combination with live face detection, has been a very active research topic recently, in particular as an analogue of fingerprint and retina recognition. But in face recognition, the approaches to solving the issue of biometric application are somewhat different. LD is the process of differentiating the space of features into animate and inanimate. Attackers will try to penetrate the system through a large number of spoof attacks, and it is LD that can significantly improve the security of facial recognition applications for biometric purposes. In face recognition, typical attack methods can be divided into several categories, their classification is based on the way in which false information is provided to the verification system, for example, a photo, a face image, a recorded video, 3D face models with the ability to blink and move lips, 3D - face models with different expressions and so on [3].

Although the best current implementations show incredible results, there is still room for growth, as well as for improving accessibility. Video quality, noise, lighting intensity strongly affect the quality of models. Also, the methods work well against known methods of attack, but are not very effective against previously unseen ones. Ultimately, one of the key issues that needs to be addressed is the availability of high-quality open datasets: their potential creation conflicts with the privacy of many people's personal data [4].

Within the framework of this work, the strategy of developing a face recognition system was chosen with an emphasis on liveness detection, which is a key element for ensuring a high level of security.

3. Development of requirements for the facial recognition system

The software should provide the following main functions (Figure 1):

Ability to capture images of 1 face at a time from the video camera for further identification in the video stream

- Ability to view already captured faces
- Ability to delete an already captured face
- Ability to change the tag name of an already captured face
- Ability to choose video stream source between webcam and video in .mp4 format
- Ability to recognize previously captured faces in the video stream with the appropriate label
- Ability to recognize unknown faces in the video stream

The given diagram of use cases reflects the most important functionality of the logical part of the application.

The software is divided into modules. Each module has its own specific set of functions. Table 1 shows the general requirements model.

The work will implement a desktop application designed for face recognition with the detection of live faces in a video stream received from a file or webcam.

The purpose of the work is to improve the identification procedure or simplify the detection of people on video for employees of the security department of enterprises by implementing face recognition methods by implementing liveness detection, increasing their accessibility due to the interface.

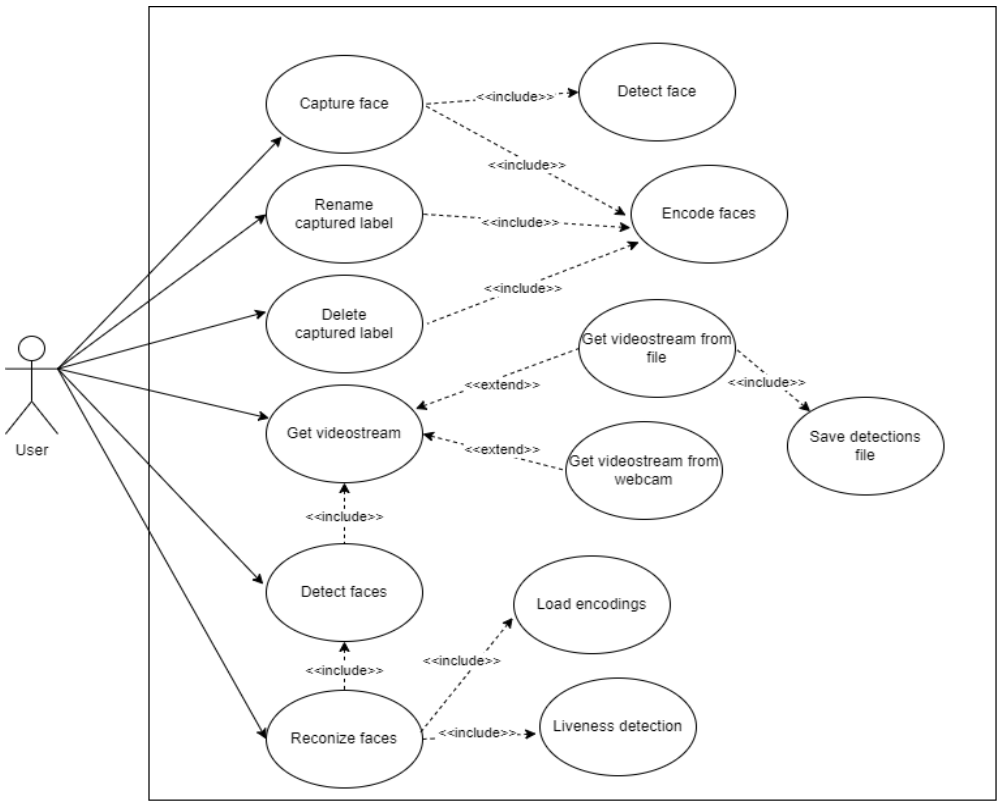

Figure 1: Use case diagram.

This goal is achieved by implementing the logical part of the program and the interface.

Table 1
General requirements model

Requirement	Priority	Risk
Capturing the face	High	High
View captured tags	Medium	Medium
Editing captured labels	Medium	Medium
Deleting captured tags	Medium	Medium
Receiving a video stream from a webcam	High	Low
Getting a video stream from a file	High	Low
Detecting faces in a video stream	High	High
Face recognition in a video stream	High	High
Saving a file with recognized faces	Low	Low
Liveness detection	Medium	High

Label format	Low	Low
Face detection	High	High

The logical part should ensure receiving the video stream, recognizing faces in it and checking them for authenticity, while the interface should show the results of the work to the user.

4. Analysis of algorithmic and technical solutions

4.1 Consider well-known algorithmic solutions for face detection.

Haarcascade (Haar Cascade Classifier) - uses Haar cascades to identify faces. It is based on the concept of "rectangles", which are simple shapes (like edges, lines) used to define faces in an image. Implementation: Training takes place using the "reinforced learning" method (AdaBoost). The system uses many different shapes and selects the best ones for effective face recognition [5].

The advantage of Haarcascade is undoubtedly speed. However, the quality of recognition at indirect viewing angles of the face leaves much to be desired. It also generates a lot of false positives, which slows down its performance.

HOG (Histogram of Oriented Gradients) - using a histogram of oriented gradients to detect faces. This algorithm divides the image into small regions and computes gradient histograms for each region to generate a facial description. HOG is often used in conjunction with a classifier such as an SVM (Support Vector Machine) to determine whether a face is present in a given region [6].

This method gives better results compared to Haarcascade, but is still not efficient enough at large angles and is more resource-intensive.

MMOD (Max-Margin Object Detection) is a deep learning (DL) method for object detection. It uses the concept of "maximum indentation" to determine the best placement of the detection box around the face. MMOD typically incorporates deep convolutional neural networks to learn facial features. It is more accurate compared to classical methods, but requires more computing power [7].

The results of testing this method were a poor ratio of detection quality to computational costs: the result of its work was much better than classical methods, but the computational complexity increased many times.

SSD (Single Shot MultiBox Detector) is another deep learning method that allows you to detect objects in images in one pass. It effectively identifies various objects and their location on the image. SSD uses a number of convolutional layers to detect objects of different sizes. This method is fast and effective, especially in real time [8]. This method showed better results compared to MMOD, while the speed was significantly better.

Each of these methods has its advantages and disadvantages, and the choice of a particular algorithm depends on the specific requirements for accuracy, speed, and computing system resources.

In the case of this development, SSD turned out to be the most effective from the point of view of the ratio of computational complexity to the quality of identification, so it will be used for the implementation of the software.

4.2 Consider face recognition algorithms.

Eigenfaces uses the method of principal components (PCA) to reduce the dimensionality of the face space. High-dimensional face data is transformed into a set of weight values that represent key facial features. The Eigenfaces implementation involves training on a set of faces to determine the principal components. Each new face is projected onto this space for comparison or recognition [9].

This method works well for systems with little data and less computing resources, but when a large number of faces and recognition quality are possible, it is of little use.

Fisherfaces uses linear discriminant analysis (LDA) to optimize the discrimination between classes of faces. They focus on maximizing the distance between face categories while minimizing the variation within each class. Fisherfaces is trained on a set of faces to determine optimal linear combinations that delineate different classes of faces [10].

This method is more efficient than Eigenfaces for situations where face classes have a high level of variability and generally has higher accuracy and computational cost.

LBPH (Local Binary Patterns Histograms) uses local binary patterns to describe facial features. This method analyzes each pixel in an image by comparing it to neighboring pixels and encodes these relationships in binary format [11].

Implementation: LBPH computes local binary pattern histograms for different face regions, and then these histograms are used for face comparison and recognition. This method is effective for variations in lighting and facial expressions.

Deep learning algorithms, mainly convolutional neural networks (CNNs), are also used for face recognition [12]. They automatically learn facial features from large datasets, discovering complex patterns and properties. They are considered to be among the most accurate methods, but require large computing resources and large amounts of data for effective training [13].

Each of these methods has its own characteristics and is suitable for different application scenarios. For example, LBPH may be a better choice for resource-constrained systems, while deep learning is ideal for complex applications with large datasets.

The CNN algorithm was chosen for the implementation for better identification quality, so it will be used for the implementation of the software.

Live face detection (LD) is an important part of biometric security systems, helping to determine whether the subject being scanned is a real person and not a photo, video or other fake. Let's consider several methods used for liveness detection [3]:

- Motion analysis boils down to distinguishing motion patterns between 3D and 2D objects, which are usually images used to trick the system. The method requires the presence of a video stream, but user interaction is not required here, which, combined with the high efficiency of 2D image recognition, is a significant advantage. Disadvantages include the need for high-quality video, as well as a decrease in efficiency if there is no active movement in the scene.
- Texture analysis takes advantage of the fact that a printed, otherwise manufactured, or pasted image will have defects, blurring, or possibly even a different environment compared to the rest of the image [14]. This approach is easy to implement and does not require user interaction, it covers a larger set of possible attacks. The disadvantages of this method are again reduced to the need for high-quality images, in addition, the efficiency drops sharply if the pattern of differences in textures was not found in the training data, and this imposes strict requirements on the volume and diversity of the dataset.

Mimic detection can be divided into two types.

- The first includes the performance of a certain movement by the user (turn the head to the right) or a certain facial expression (smile). If this instruction is fulfilled, then the verification is successful. This is the only method considered that requires active interaction with the user, and, accordingly, the disadvantage is the introduction of the human factor.
- The second category consists in the detection of facial expressions characteristic of all living people, for example - blinking of the eyes. The complication that this method brings is the need to additionally detect certain parts of the face, such as eyes or something. In general, mimicry is more effective than other methods, but requires additional means of implementation.

Analysis of textures and patterns was chosen for implementation. Since there are no ready-made open solutions, an own CNN network will be developed, due to the simplicity of implementation and good results [15], which it shows in comparison with similar models.

So, SSD model for face detection and CNN model for recognition and detection of living faces were chosen for development. This creates high requirements for the hardware so that the process can take place in real time.

5. Software architecture

For front-end application development, it is important to separate views, logic, and data to create a more flexible and modular architecture. However, considering that all the data that the application operates on is a person's label and an image for its identification, and one of the functional requirements is the ability to capture a new face for identification, it is better to store this data locally than in a separate database. Taking into account the peculiarities of the Python language and the large number of libraries, the elements of the Adapter and Decorator (Wrapper) patterns will be used.

Consider the component diagram (Figure 2). Since the Qt library is designed for what-you-see-is-what-get interface development, the use of other architectural patterns is not appropriate.

The interface will look like a desktop application implemented using PyQt6, with the main screen, video stream windows, and dialog windows, which will give the user the opportunity to access all the functionality of the software, as well as enter the necessary data and see the result of the program.

The business logic will load face encoding, models for FD, FR and LD, as well as receive a video stream, which will be processed in a separate thread from the rest of the tasks. OpenCV capabilities will be used to process, retrieve, save, and output frames.

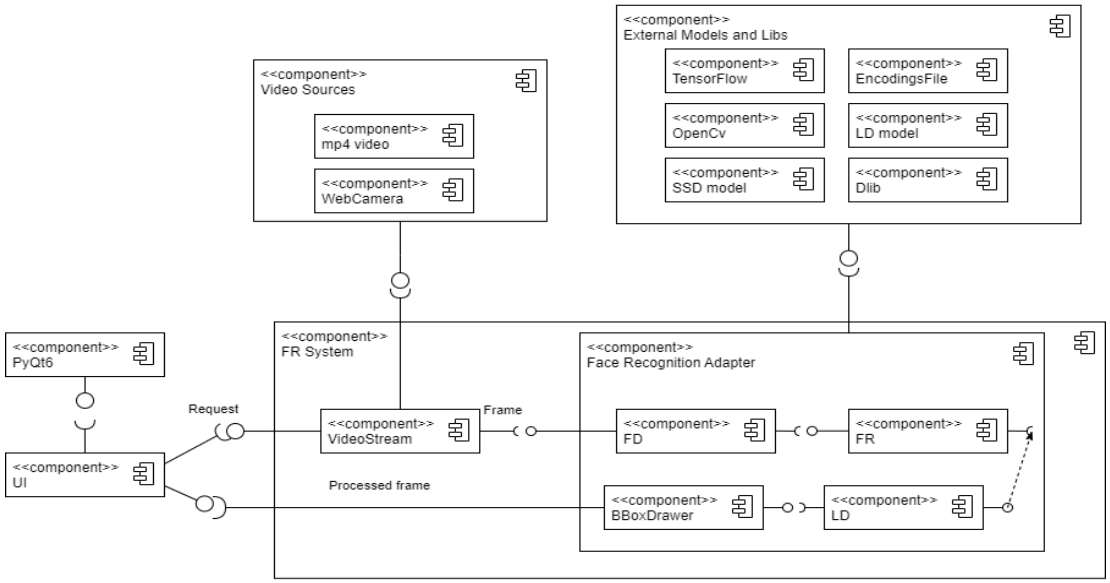

Figure 2: Diagram of components.

The data on each captured face (20 images) will be stored in a separate folder, and the characteristics will be extracted for recognition and encoded into a 128-length vector using the Dlib library.

5.1 Description of used algorithms and architectures of neural networks

To create a video stream, OpenCV capabilities were used, which allows you to capture a video stream from both webcams and video. Because real-time processing requires fast access to stream frames with minimal latency, the Tread module was used. The stream is constantly updated in parallel with the processing of the previously taken frame, which significantly reduces delays.

To find faces in a frame from a video stream, a neural network of the SSD architecture was used. The SSD approach is based on a convolutional network that creates a collection of BBoxes and estimates of the presence of class object instances within those frames. BBoxes are similar to those used in Faster R-CNN, so they take into account the aspect ratio for a rectangle with an object, so they take into account the aspect ratio, but apply to feature maps at a different scale. The neural network used was trained for face recognition in the Caffe framework, so it can be loaded using the DNN module of the OpenCV library. This model combines high quality with good performance, allowing you to recognize faces from different angles. The detailed architecture of the layers of the neural network can be seen in figure 3.

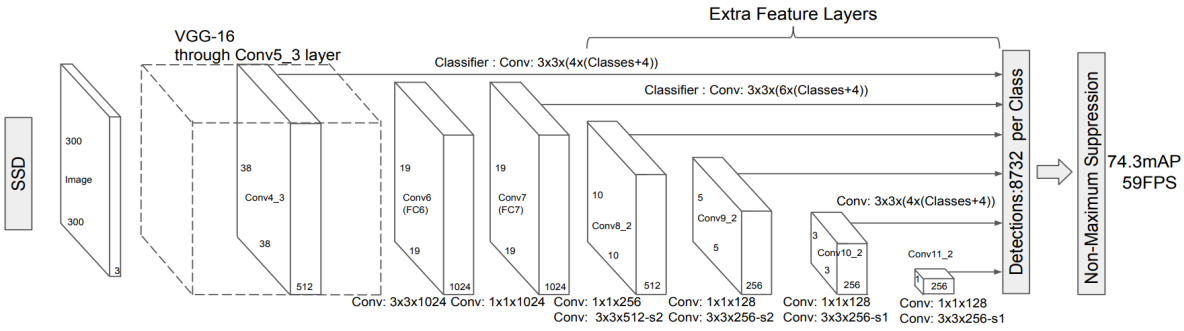

Figure 3: SSD model architecture.

For face recognition, the capabilities of the Dlib library are used. Using MMOD (Max-Margin Object Detection) [7] detects 68 points [17] on the face, after which they will be encoded into a feature vector of length 128 using the ResNet model, which makes it possible to compare the image with the previously saved one - finding the difference encodings, we get the distance between the images, and the smaller it is, the more likely it is one person [17]. The face_recognition package provides a convenient interface for such a solution built on the basis of Dlib.

A proprietary neural network was trained for LD. Given the high efficiency and speed of CNN models in this problem [16], this architecture was chosen. The task was considered as a binary classification. The input training image was 150 by 150 pixels. TensorFlow and Keras libraries were used for training. Figure 4 shows the resulting architecture.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 17, 17, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 15, 15, 128)       147584

 max_pooling2d_3 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 512)               3211776

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Architecture of the LD model.

The output layer contains an estimate of the probability that the image contains a live face. The Adam algorithm was used for optimization.

5.2 Description of the dataset

For face recognition, it is necessary to save the label assigned to the person and his image. The greater the number of different types of images, the better the recognition quality, but the longer the capture process. Therefore, the number of images was chosen to be 20. Since continuous access to the images is required to display in the interface and encode the images and simultaneously be able to capture them in real time, it was chosen to store them locally.

The images of each captured face are stored in a corresponding folder and with a corresponding label. The os library is used to edit, modify and access them.

The schedule of the learning process can be seen in Figure 5.

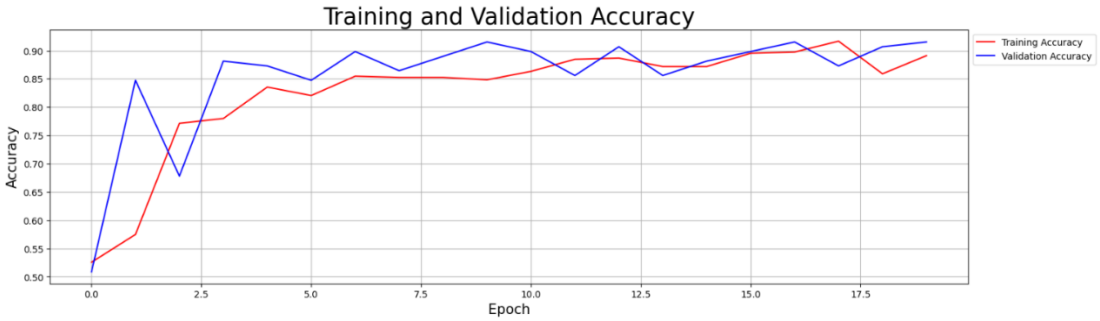

Figure 5: Learning graph of the LD model.

To train a neural network for LD, you need a labeled dataset of 2 types of labeled data - real and fake faces. The data was obtained by recording a video of a face (real) and recording a video shown from a phone to another camera (inanimate). In addition, data was taken from the CelebA-Spoof dataset, and during training, their augmentation took place. Thus, a large number of possible attacks are covered: from the demonstration of a video recording or a single image to the creation of fake (printouts, photos, 3d models) faces.

6. Ways of improvement and further development

We can distinguish 3 main directions of further development:

1. Implementation of the process of integration and deployment of the application on the server;

Improving the efficiency of the application with the ability to work in real time on devices without GPU;
Improvement of LD.

For the first, it is necessary to place the application on a server that will have access to webcams, as well as to integrate interaction with an external database.

For the second direction, a deeper use of parallel programming technologies is necessary, as well as possible replacement of the used methods with lower quality, but more optimal from the point of view of computing costs.

The third direction is to develop a dataset that covers more attack types, covers more LD approaches, and improves neural network architecture.

7. Conclusions

As a result of the research, a desktop application designed for detection, recognition and detection of live faces in a video stream was designed. The result of the development is the improvement of the identification procedure or the simplification of the detection of persons on video for employees of the security department of enterprises by implementing face recognition methods by implementing liveness detection, increasing their accessibility due to the interface.

Python is chosen as the main development environment. PyCharm IDE and Jupyter-notebook were chosen as development environments. Many libraries were used to implement the project, including Keras and Dlib for neural networks, OpenCV for video streams and image processing, and Qt for the interface.

Architectural solutions were analyzed and architectural patterns were selected. Neural network architectures, known algorithmic solutions were also analyzed and the optimal ones for the given task were selected from among them.

After analyzing the known successful software products, niches were identified that require the presentation of a new solution. Based on them, functional and non-functional requirements were developed, and the problem statement was made.

Then an analysis of the business processes of the software was carried out, architecture planning and testing of the software for compliance with the requirements were implemented.

The novelty of the work consists in the modification of the face recognition process in the video stream by implementing our own Liveness Detection model.

The practical significance of the developed system is as follows:

The facial recognition system allows you to automate the processes of entrance control and attendance, ensuring safe and convenient access for employees.

The use of the facial recognition system helps to increase the level of security on the territory of the enterprise, prevents unauthorized access and helps to respond to events in a timely manner.

The system can be used to accurately record the time of arrival and departure of employees, which facilitates the accounting of working hours and improves control over the work schedule.

The facial recognition system allows quick and efficient identification of persons, which leads to a reduction in registration and verification time.

The obtained results solve the set tasks and achieve the specified goal, which were described in the technical task. However, there is still room for improvement, especially in LD performance and quality.

References

[1] M. Basurah, W. Swastika, O. H. Kelana (2023). Implementation of face recognition and liveness detection system using tensorflow.js. Jurnal Informatika Polinema. 9. 509-516. DOI:10.33795/jip.v9i4.1332.

[2] S. Popereshnyak, O. Suprun, O. Suprun and T. Wieckowski, "IoT application testing features based on the modelling network," 2018 XIV-th International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH), pp. 127-131, doi: 10.1109/MEMSTECH. 2018.8365717.

[3] S. Chakraborty, D. Das (2014). An Overview of Face Liveness Detection. International Journal on Information Theory. 3. 10.5121/ijit.2014.3202.

[4] S. Khairnar, S. Gite, K. Kotecha, S. D. Thepade (2023) Face Liveness Detection Using Artificial Intelligence Techniques: A Systematic Literature Review and Future Directions / Big Data Cogn. Comput. 7(1), 37. DOI:10.3390/bdcc7010037

[5] C. H Choi, J. Kim, J. Hyun, Y. Kim, B. Moon (2022). Face Detection Using Haar Cascade Classifiers Based on Vertical Component Calibration. Human-centric Computing and Information Sciences, 12, Article 11. DOI: 10.22967/HCIS.2022.12.011

[6] C. Rahmad, R. A. Asmara, D. R. H. Putra, I. Dharma, H. Darmono, I. Muhiqqin (2020) Comparison of Viola-Jones Haar Cascade Classifier and Histogram of Oriented Gradients

(HOG) for face detection/ IOP Conference Series: Materials Science and Engineering, Volume 732, DOI 10.1088/1757-899X/732/1/012038.
[7] D. E. King Max-Margin Object Detection. URL: https://arxiv.org/ abs/1502.00046
[8] M. Turk, A. Pentland Eigenfaces for Recognition. Journal of Cognitive Neuroscience (1991) 3 (1): 71–86. https://doi.org/10.1162/jocn.1991.3.1.71
[9] M. Anggo, L. Arapu (2018) Face Recognition Using Fisherface Method. Journal of Physics Conference Series 1028(1): 012119
[10] F.Deeba, H. Memon, F. Ali, A. Ahmed, A. Ghaffar, (2019). LBPH-based Enhanced Real-Time Face Recognition. International Journal of Advanced Computer Science and Applications. 0.14569/IJACSA.2019.0100535.
[11] When Face Recognition Meets With Deep Learning: An Evaluation of Convolutional Neural Networks for Face Recognition. URL: https://www.cv-foundation.org/openaccess/content_iccv_ 2015_workshops/ w11/html/ Hu_ When_Face_Recognition_ICCV_2015_paper.html
[12] Popereshnyak, S., Vecherkovskaya, A., Zhebka, V. Intrusion Detection based on an Intelligent Security System using Machine Learning Methods. CEUR Workshop Proceedings, 2024, 3654, pp. 163–178
[13] R. Koshy, A. Mahmood (2019) Optimizing Deep CNN Architectures for Face Liveness Detection. Entropy 2019, 21(4), 423. DOI: 10.3390/e21040423
[14] Chebanyuk, O. Multilingual Question-Driven Approach and Software System to Obtaining Information from Texts / CEUR Workshop ProceedingsThis link is disabled., 2022, 3501, pp. 256–265
[15] Top 11 Facial Recognition Software in 2021. URL: https://www.spiceworks.com/it-security/identity-access-management/articles/facial-recognition-software/face_recognition package.
[16] Dlib C++ library. URL: http://dlib.net/python/ index.html#dlib_pybind11. face_ recognition _model_v1
[17] CelebA-Spoof: Large-Scale Face Anti-Spoofing Dataset with Rich Annotations. URL: https://github.com/ZhangYuanhan-AI/CelebA-Spoof