Development of a user authentication module based on computer vision and machine learning technologies

Ivan A. Hrytsai[1], Vasyl P. Oleksiuk[1,2] and Yaroslav P. Vasylenko[1]

[1]*Ternopil Volodymyr Hnatiuk National Pedagogical University, 2 Maxyma Kryvonosa Str., Ternopil, 46027, Ukraine*
[2]*Institute for Digitalisation of Education of the NAES of Ukraine, 9 M. Berlynskoho Str., Kyiv, 04060, Ukraine*

Abstract

This paper addresses the development of a user authentication module based on computer vision and machine learning technologies. The study analyses traditional authentication methods, including knowledge-based, possession-based, and biometric approaches, identifying their respective limitations in the context of contemporary cybersecurity threats. A comprehensive comparison of modern libraries for biometric data processing—including OpenCV, Dlib, Face Recognition, DeepFace, and InsightFace—is presented, with particular attention to their accuracy, performance, and implementation complexity. The proposed authentication module employs facial recognition technology implemented using Python with OpenCV, dlib, and face_recognition libraries, leveraging a pre-trained ResNet-34 deep convolutional neural network for generating 128-dimensional facial embeddings. The system architecture integrates a web-based user interface, PHP-based server logic, and a Python image processing subsystem, enabling real-time biometric verification. Additionally, the module incorporates dynamic head movement detection using MediaPipe FaceMesh to enhance security against spoofing attacks. Functional testing utilising LFW and CelebA datasets demonstrated identification accuracy exceeding 96.8%. The results confirm the effectiveness of the proposed approach for practical implementation in information security, education, and access control systems.

Keywords

authentication, machine learning, facial recognition, biometrics, OpenCV

1. Introduction

Personal data protection is one of the top priorities for modern information systems. User authentication is the process of confirming their identity, which is usually based on passwords, tokens, or multi-factor verification. However, with the development of digital technologies and the growth of cyber threats, traditional methods are becoming less effective, giving way to biometric solutions, in particular, machine learning-based facial recognition. The use of artificial intelligence allows for some factors to be taken into account, such as facial expressions, changes in lighting, and head rotation angles. It significantly increases the accuracy and reliability of authentication. So, the application of machine learning in authentication tasks opens up new opportunities for building secure, scalable, and flexible information systems that can adapt to changes and withstand modern threats.

The purpose of this article is to analyse existing approaches, design an authentication module based on machine learning, and implement face recognition technology. To achieve this goal, the following tasks have been set:

1. Review and classify authentication methods.
2. Selection of the technologies for the development of the authentication module.
3. Design architecture and develop the authentication module.
4. Training and validation of machine learning model.

CS&SE@SW 2025: 8th Workshop for Young Scientists in Computer Science & Software Engineering, November 21, 2025, Kryvyi Rih, Ukraine
https://doi.org/10.55056/cssesw2025/paper14
grytsaj_ia@fizmat.tnpu.edu.ua (I. A. Hrytsai); oleksyuk@fizmat.tnpu.edu.ua (V. P. Oleksiuk); yava@fizmat.tnpu.edu.ua (Y. P. Vasylenko)
https://tnpu.edu.ua/faculty/fizmat/oleksyuk-vasil-petrovich.php (V. P. Oleksiuk)
0000-0001-6825-4697 (V. P. Oleksiuk); 0000-0002-3121-7005 (Y. P. Vasylenko)

2. Basic approaches to user authentication

Data protection technologies are constantly evolving, and modern authentication methods are becoming more reliable and convenient. In recent years, many innovative solutions have emerged that combine a high level of security with ease of use. Scientific literature discusses authentication methods such as:

- *Knowledge-based authentication.* Such methods are based on the user knowing the authentication data, such as passwords, PIN codes, and answers to secret questions. Password authentication is the most common, simple, and familiar method. In this case, when the subject enters their password, the authentication subsystem compares it with the password stored in the reference database in encrypted form. This method of identification has some significant drawbacks, such as the password can simply be forgotten, stolen, or passed on to another person. If the password is too simple, it is easy to guess. Short passwords are easy to crack using brute force. Passwords with a large number of characters will be more resistant to cracking, but they are inconvenient to use and difficult for users to remember [1].
- *Possession-based authentication.* These methods are based on the user having an authentication device, such as a smart card, token, or smartphone. Such a device is usually difficult to reproduce or forge, which makes such methods more secure than simple knowledge-based methods. The advantages of this method are higher security compared to passwords and resistance to phishing. Unfortunately, the loss of the device makes authentication impossible [2].
- *Biometric-based authentication* is based on unique physiological or behavioural characteristics. This group of methods involve the use of equipment to measure and compare the user's specified individual characteristics with a reference standard. Biometrics is a set of automated methods for authenticating people based on their physiological (static) and behavioural (dynamic) characteristics. Physiological characteristics include fingerprint, retina, and cornea patterns, hand and face geometry, etc. Behavioural characteristics include signature dynamics, keyboard style, and voice recognition. Such means allow for high-accuracy recognition of the owner based on a specific biometric feature, and it is practically impossible to falsify such parameters, but they raise certain privacy issues [3]. Apple Face ID is one of the most powerful biometric authentication technologies. Unlike 2D methods, which are based on analyzing a flat image, this method projects physical light into the surrounding environment. Such systems do not passively "see" faces, they actively emit their own structured light in the near-infrared range with a known pattern of near 30000 infrared dots. By measuring how this pattern of dots is distorted when it hits the curves of the user's face, the system calculates an accurate 3D depth map [4].
- *Hybrid methods*, or so-called multi-factor (two-factor) authentication (MFA, 2FA), combine a password with a one-time code (for example, via SMS or a mobile app). This is one of the most effective ways to protect accounts of email, messengers, social media accounts, and others. The method significantly increases protection, but creates dependence on third-party devices. The disadvantages of these methods are the complexity of implementation or a possible reduction in user convenience [5].

Among the methods listed, biometric authentication is particularly relevant for application development. This is due to its ability to combine high security, convenience, and compatibility with modern technologies. Currently, this method is used in such areas as:

- Security, where such systems are used for contactless entry into buildings and offices, monitoring employee attendance, as well as in smart surveillance cameras for automatic identification of individuals [6, 7].
- Healthcare, where biometric authentication is used to identify patients and medical staff, preventing errors when accessing electronic medical records and ensuring data privacy.
- E-government, where facial recognition technologies are used for remote verification of citizens when providing administrative services or voting. BankID and MobileID systems are used for this purpose.

- Modern mobile applications are also massively switching to biometric authentication. Almost every modern smartphone has a camera and a sensor that allows authentication by face or fingerprint. These technologies are used to confirm payments, in social media and messenger apps (biometric access lock option for correspondence), and in corporate VPN apps for two-factor protection.
- In online banking, where biometrics is becoming a cornerstone of security, many banking apps allow you to log in with your fingerprint or face instead of a password, which is not only convenient but also reduces fraud.

Each of these methods has its advantages and disadvantages. However, none of them confirm the authenticity of a specific subject; they only record the fact that the subject's authenticator matches its identifier. That is why all known methods are vulnerable to authenticator compromise [8].

3. Using machine learning for authentication

Machine learning, particularly deep learning, opens up new possibilities in the development of authentication systems [9, 10, 11, 12, 13, 14, 15, 16, 17]. Applications developed using these technologies are capable of adapting to changes in input data and independently improving their accuracy over time. While classical recognition algorithms [18, 19] use manually designed features (Haar cascades or local binary patterns) and simple classifiers, modern approaches automatically learn features from large datasets of face images. Let's briefly describe the approaches and models used for facial recognition for authentication purposes.

Histogram of Oriented Gradients (HOG). This is one of the basic computer vision algorithms, proposed back in 2005. It analyses the statistics of brightness gradients in image fragments. HOG is successfully used for face detection. In particular, the Dlib library implements a face detector with subsequent classification using the SVM support vector method. Its advantage is high performance and real-time operation [20]. However, the accuracy of these methods drops sharply for non-standard angles or partially covered faces. For direct recognition, HOG was also used in combination with classifiers, but it is inferior in accuracy to modern deep models.

Deep Neural Networks (DNN) are the multi-level models capable of detecting hidden patterns in large amounts of data [21]. They are used both for classifying users by biometric features and for constructing stable feature vectors (embeddings) that describe individual characteristics of the face, iris or fingerprints [22]. Among DNNs, a special place is occupied by *Convolutional Neural Networks (CNN).* They detect and hierarchically encode characteristic facial features (eye position, jaw shape, eyebrow lines), converting images into vectors of spatial features. These vectors allow the system to calculate the degree of similarity between faces in high-dimensional Euclidean space [23]. The advantage of CNN is that it automatically takes into account variations caused by changes in lighting, posture, and facial expression. This is possible thanks to training on large samples, which significantly increases the reliability of authentication in real-world conditions. The Facebook team demonstrated the first breakthrough results in the DeepFace method [24], achieving an accuracy of about 97%. Google's FaceNet architecture, based on a 22-layer Inception neural network, achieved an even higher accuracy record of 99.63% on the LFW (Labeled Faces in the Wild) dataset [25]. It surpassed human performance in face verification tests for the first time. Open implementations of these models appeared as early as 2015–2016. One of them was the OpenFace project. It provided access to deep face recognition in open source code, implemented in the face_recognition library. The use of data augmentation methods such as lighting variations, rotation, scaling, noise modelling, or facial expressions can significantly increase the number of training examples and improve the model's ability to generalise [26]. This is especially important for real-time data processing or edge systems, where the system must be resistant to sudden changes in environmental conditions, camera angles, and facial expressions [27].

ResNet with additional loss mechanisms. It is one of the most effective models for faces, ArcFace, which was built on the basis of the ResNet-100 architecture. It uses Additive Angular Margin Loss, which increases the difference between embeddings of different faces, maximising the angular distance

between classes [28]. This made it possible to obtain highly discriminative features and set new accuracy records. According to the results [29], the ArcFace model on ResNet-100 achieved 99.83% accuracy on the LFW dataset and outperformed its predecessors on more complex tests. The implementation of ArcFace in the open InsightFace library has made this technology an industry standard: the InsightFace platform supports training models with ArcFace and is used in many real-world systems, providing the highest face recognition accuracy of 95–99% [30]. The disadvantage of deep ResNet models is their computing resource requirements. Powerful graphics processors and large arrays are required for their training. Therefore, hardware accelerators are often required for real-time deployment. Despite this, optimised models (with fewer parameters or quantisation) are capable of running on mobile devices, albeit with some reduction in accuracy. Currently, there are approaches such as SphereFace, CosFace, and MagFace, which offer alternative loss functions or architectural improvements to increase recognition accuracy [31].

Transformers and modern hybrid models. A new stage in face recognition was the introduction of Vision Transformer (ViT) architectures. Initially, it was believed that such models required very large amounts of data and would be slower than CNN, but recent studies refute this [32]. The results show that transformers outperformed CNNs in terms of accuracy and robustness. So the ViT model showed better resistance to complications such as partial face occlusion by masks or large distances to the camera. At the same time, the performance of transformers proved to be competitive. ViT inference time is only 24% longer than that of the fastest CNN-MobileNet, and the memory required for ViT is less than some convolutional networks. This is achieved through efficient parallelism and the absence of cumbersome convolutional layers. Some works propose hybrid approaches combining CNNs for local feature extraction and transformers themselves for global face recognition [33].

An analysis of papers indexed by Scopus over the past five years confirms the growing interest in user authentication using machine learning, in particular computer vision technologies, face recognition, and OpenCV libraries. Thus, Divya et al. [34] have presented a real-time identification system using OpenCV and YOLOv8 for face detection and recognition, integrated with Django for web-based administration. The system processes live video feeds, employing YOLOv8 for precise face detection and OpenCV for frame processing. It achieves scalability and efficiency, aiding law enforcement in rapid identification, though the paper is not open access. Saadat et al. [35] and his co-authors proposed a real-time face detection and identification system for networked video surveillance, utilizing OpenCV's image processing capabilities. Tested on IP cameras and Raspberry Pi, the system demonstrates simplicity and effectiveness, achieving more than 95% face detection success rate and 80% recognition accuracy.

Face Spoofing Detection is another direction of research in exported metadata. developed the FCN-DA-LSA method for face spoofing detection, combining fully convolutional networks with domain adaptation and lossless size adaptation [36]. This study emphasises detecting spoofing artefacts in face recognition systems. Indian scientists explored gender classification using facial embeddings extracted via a pretrained Inception Network, achieving also more than 97% accuracy. Results demonstrate the effectiveness of deep learning in classifying biometric traits, with potential applications in personalised authentication systems [37].

The authors of the analysed papers pay considerable attention to choosing the development tools. Modern libraries are conventionally divided into three categories such as

- universal computer vision libraries (OpenCV);
- specialized solutions for working with faces (Dlib, Face Recognition);
- high-level frameworks based on deep learning (DeepFace, InsightFace).

Table 1 provides a comparative overview of the main libraries.

As can be seen from the table 1, the InsightFace library offers the highest precision with greater hardware demands. At the same time, OpenCV excels with its fast processing and large community support. It leverages the Haar and DNN models, but has some limitations in recognition accuracy. Given its versatility, robust community backing, and suitability for real-time applications, we chose OpenCV for developing a biometric authentication module.

Table 1
Comparison of face recognition libraries and frameworks.

Library	Language	Advantages	Disadvantages	Model support
OpenCV	C++, Python	Fast processing, large community	Limited accuracy in recognition	Haar, DNN
Dlib	C++, Python	High precision, support HOG/CNN, face alignment	Slow work with large data sets	CNN, HOG
Face Recognition	Python	Simple API, Accurate comparison	Less flexibility, dependency on Dlib	Dlib
DeepFace	Python	Multi-model support, visualization	High resource requirements	VGG-Face, FaceNet, OpenFace
Mediapipe	C++, Python	Fast tracking, mobile optimization	Does not support identification	Face Mesh, Detection
InsightFace	Python	Highest precision, ArcFace, industrial use	High hardware requirements	ArcFace

4. Development of an authentication module

Research suggests that developing a biometric authentication module using face recognition involves collecting facial data, preprocessing it, training a model, and testing for accuracy. Let us briefly describe the stages of development (figure 1).

The first steps of the development process are the diagram highlights algorithm selection, library selection, and programming language choices. After that, data acquisition is performed, where user images obtained from cameras in real time are used for facial authentication. The next step is cleaning and preparing the data through normalization, noise reduction, cropping, resizing, and alignment to ensure quality. This step is essential for creating a reliable dataset for training. Next, a model is trained to recognize unique facial features, often using libraries like OpenCV. The obtained face recognition model is capable of identifying or verifying a user by comparing feature vectors. It must be tested for accuracy, using real-world devices (web camera in our case). Algorithm selection involves choosing appropriate machine learning or computer vision algorithms, such as Eigenfaces, Fisherfaces, or CNNs, based on the specific requirements of the face recognition task.

The OpenCV library, which contains built-in high-performance algorithms for object detection and identification, is used to implement the recognition system. OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning library. The cross-platform library focuses on real-time image processing and includes unlicensed implementations of the latest computer vision algorithms. OpenCV is released under the BSD license, which allows the library to be used for both commercial and research purposes.

The face recognition software system implementation uses the high-level face_recognition library, which is based on dlib functionality. This library provides a complete image processing cycle: from face detection to vector representation and embedding comparison. The basis for image vectorisation is a deep convolutional neural network of ResNet-34 architecture, implemented as part of dlib. The model was pre-trained on a large set of face images and allows obtaining 128-dimensional feature vectors (face embeddings), which are an informative representation of each face. This allows for accurate comparisons between images regardless of rotation angle, facial expressions, or lighting conditions.

The recognition stage itself involves performing tasks such as:

- scaling the input image;
- detecting faces using an HOG detector or a CNN convolution detector;
- constructing a vector representation of the face using a pre-trained model;
- calculating the Euclidean distance between the obtained vector and vectors from the image database;

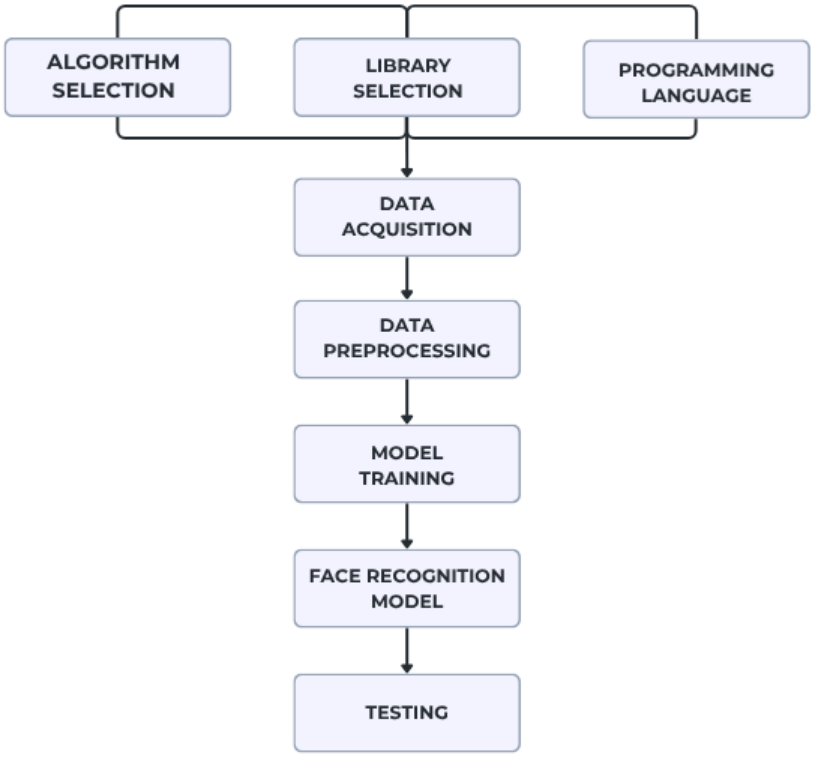

Figure 1: Stages of developing a biometric authentication module.

- making a decision about the identification of a person based on a similarity threshold.

The developed biometric authentication module is implemented as a multi-level system that combines a web user interface, server logic in PHP, and an image processing subsystem in Python. The main idea behind modular design is to distribute functions among system components. PHP accepts incoming HTTP requests, provides communication with the user interface, processes images in Base64 format, and initiates Python script calls. All resource-intensive operations related to computer vision and the construction of vector facial features are processed by Python, which has extensive capabilities for implementing machine learning and image processing algorithms.

The logic of the authentication module is as follows: the user submits a photograph of their face as an image via the web interface (figure 2). Than it is sent to the PHP processor; the image is encoded into a Base64 string, decoded, and transmitted as a parameter in the command line when the Python script is launched; the script performs face detection, vector representation (face encoding), comparison of the resulting vector with existing templates in the system, or saving a new template as a JSON file.

This approach provides flexibility, scalability, and adaptability of the solution to various deployment scenarios, both in small local networks and on more powerful cloud servers [38]. The physical separation of web interface and the machine learning module allows the system to be scaled independently in the future, For example, it may be for implementing a Python service as a separate RESTful API).

4.1. Basic methods of the biometric module

The module's algorithm provides for the implementation of such functions as new user registration and authentication based on a person's image. The architecture of the module is as shown in figure 3.

Below we describe the key methods that implement the logic of its operation.

The Register component was created to create a new user account (see listing 1).

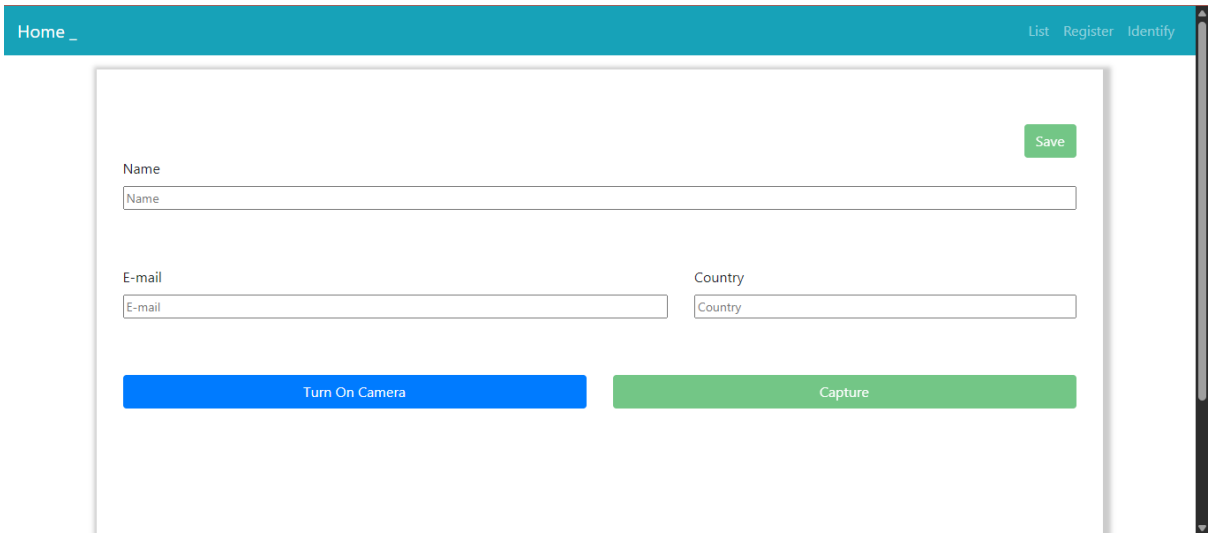

Figure 2: Web interface for user registration.

Listing 1: Creating component for user registration.

```
export default {
    name: "Register",
    mixins: [validationMixin],
    validations: {
        name: { required },
        email: { required, email },
    },
```

This fragment declares a component named 'Register'. The component connects validationMixin, which provides calls to validation methods from the Vuelidate library. The validations object specifies that the name field is mandatory, and the email field is additionally checked for compliance with the email format. Thanks to this part, the form components respond to user input and can block its submission if there are errors. In the context of biometric authentication, this block ensures that user data such as name, country, and email is set before video analysis is launched. Validation is reactive, meaning it changes in real time. Validation does not affect face tracking. This structure allows you to avoid basic input errors before video authentication begins.

During registration, the user provides an image of their face (figure 4), which is processed by a Python module. Using the face_recognition library, which in turn uses Dlib and a pre-trained ResNet neural network, the coordinates of the face in the image are detected. After that, a 128-dimensional face encoding vector is formed, which uniquely represents the spatial features of the person's appearance. This vector is serialised in JSON format and stored in the corresponding directory with the user ID.

Biometric face processing begins with the initialisation of the MediaPipe FaceMesh model (listing 2).

Listing 2: Initialisation of reactive data.

```
async initFaceMesh() {
  this.faceMesh = new FaceMesh({
    locateFile: (file) => `https://<URL>/face_mesh/${file}`,
  });

  await this.faceMesh.setOptions({
    maxNumFaces: 1,
    refineLandmarks: true,
    minDetectionConfidence: 0.5,
```

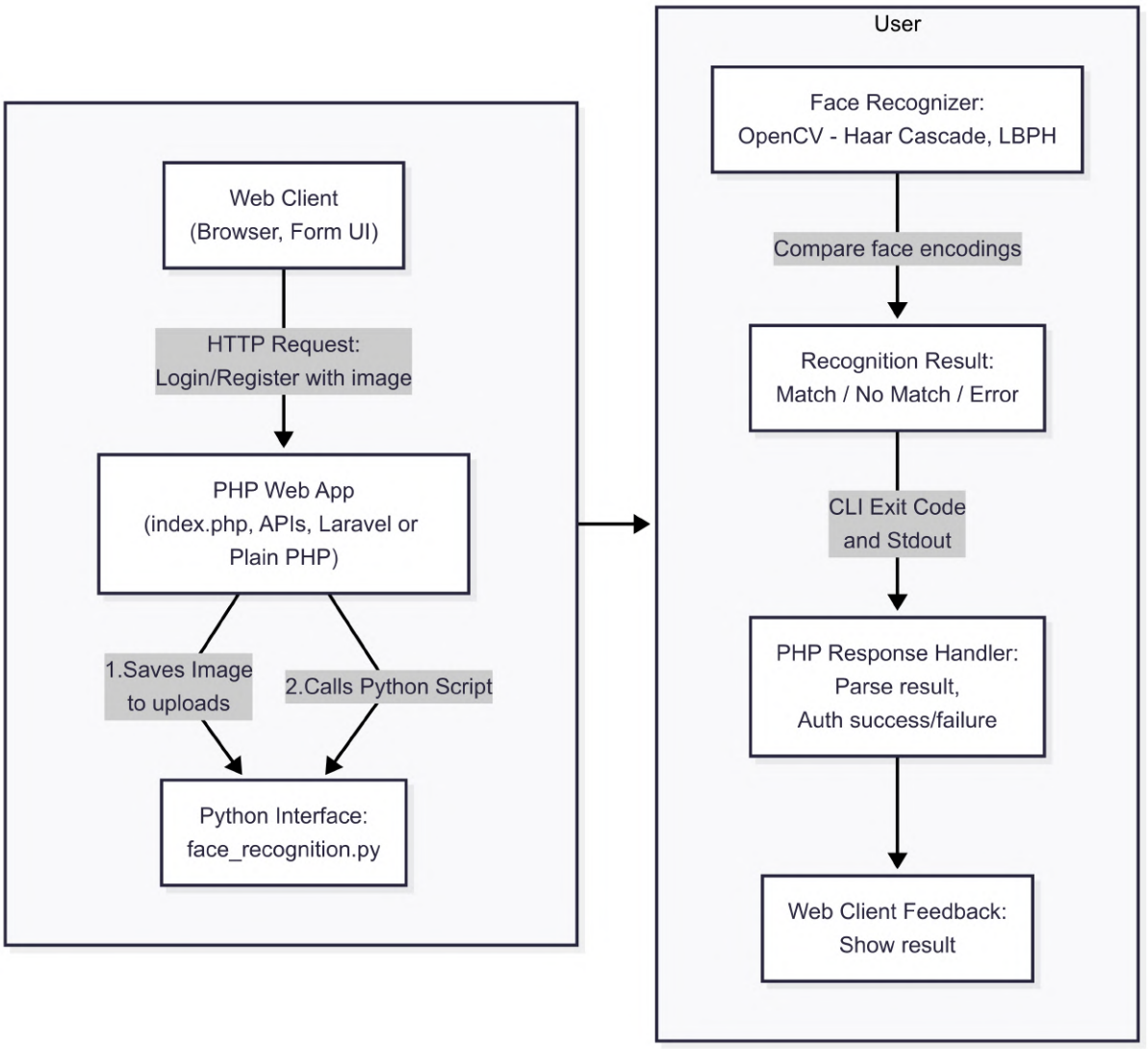

Figure 3: The architecture of the module.

```
    minTrackingConfidence: 0.5,
  });

  this.faceMesh.onResults(this.onResults);
},
```

The function creates an instance of the FaceMesh model to determine key points on the face. The locateFile method allows you to load the model from CDN. Next, the parameters are set. For example, *maxNumFaces: 1* tracks only one face; *refineLandmarks: true* is an advanced definition of key points for pupils, eyes, and lips; *minDetectionConfidence* and *minTrackingConfidence* are detection/tracking confidence thresholds. These parameters determine the accuracy of the analysis. *onResults()* is a method that is called after each processed frame and initialises the neural network.

The processing of the obtained data in the FaceMesh model is carried out as follows (listing 3).

Listing 3: Processing of the image.

```
onResults(results) {
 if (results.multiFaceLandmarks.length > 0) {
   const landmarks = results.multiFaceLandmarks[0];
   this.detectHeadMovement(landmarks);
```

Figure 4: Image capture during user registration or authentication.

```
    this.drawLandmarks(landmarks);
  }

  this.faceMesh.onResults(this.onResults);
```

The onResults() method is called after each frame is processed in FaceMesh. If a face is found (condition multiFaceLandmarks.length > 0), the system receives an array of key points (landmarks). Next, two methods are called

- detectHeadMovement(), which analyses the position of the user's head (left, right, tilt, etc.);
- drawLandmarks(), which visualises these points on the canvas.

In this way, the method coordinates the process between analysis and visualisation. It runs several times per second, depending on the performance of the device. If no face is detected, the frame is ignored.

The detectHeadMovement method was created to detect the position of the user's head (listing 4).

Listing 4: Detecting the position of the user's head.

```
detectHeadMovement(landmarks) {
  const nose = landmarks[1];
  const leftEar = landmarks[234];
  const rightEar = landmarks[454];
  const noseX = nose.x;
  const leftEarX = leftEar.x;
  const rightEarX = rightEar.x;
  const headTiltX = rightEarX - leftEarX;
  if (noseX < 0.4) {
    this.headMovement = "Turn Right";
```

```
    } else if (noseX > 0.6) {
      this.headMovement = "Turn Left";
    } else {
      this.headMovement = "Head Straight";
    }
    const noseY = nose.y;
    if (noseY > 0.65) {
      this.headMovement = "Tilt Down";
    } else if (noseY < 0.40) {
      this.headMovement = "Tilt Up";
    }
}
```

The method uses the coordinates of key points on the face: the nose and ears. If the horizontal x-coordinate of the nose is less than 0.4, the head is turned to the right, and the nose shifts to the left in the video. If the value is greater than 0.6, it is turned to the left. Between them, the position is considered straight. Similar values (0.65 and 0.4) are used to detect vertical movements. Head movements can be used to confirm authenticity. For example, the user may be asked to tilt their head or turn it to the side to verify that it is a real person and not an image. This makes the authentication process more secure and aimed at preventing fraud (for example, when using a photo).

The drawLandmarks function performs a graphical representation of key points that were defined by the FaceMesh model (listing 5).

Listing 5: Detecting the position of the user's head.

```
drawLandmarks(landmarks) {
  const canvas = this.$refs.overlayCanvas;
  const ctx = canvas.getContext("2d");
  ctx.clearRect(0, 0, canvas.width, canvas.height);
  ctx.strokeStyle = "red";
  ctx.lineWidth = 1;
  landmarks.forEach((point, index) => {
    const x = point.x * canvas.width;
    const y = point.y * canvas.height;
    ctx.fillStyle = index === 1 ? "blue" : "lime";
    ctx.beginPath();
    ctx.arc(x, y, 2, 0, 2 * Math.PI);
    ctx.fill();
  });
}
```

First, the canvas is cleared (clearRect). This is done to avoid overlapping frames. Then, for each point, the position on the canvas is determined according to its normalised coordinates (x, y values within [0, 1]). Special attention is paid to the point with index 1. This is the nose. The point is marked in blue. All other points are drawn in green. This allows the developer or researcher to visually assess the accuracy of the tracking and can be used as a debugging tool. Visualisation helps to ensure that the system is correctly tracking key areas of the face in real time.

The captureImage method is used to save an image from a video stream (listing 6).

Listing 6: Saving the image from camera.

```
captureImage() {
  let snapshotCanvas = this.$refs.snapshot;
  let context = snapshotCanvas.getContext("2d");
  const player = this.$refs.player;
```

```
  snapshotCanvas.width = player.videoWidth;
  snapshotCanvas.height = player.videoHeight;
  context.drawImage(player, 0, 0, snapshotCanvas.width,
  snapshotCanvas.height);
}
```

A separate canvas element is used, where the current image from the webcam (player) is transferred using drawImage(). Before this, the dimensions of the snapshot canvas are set to match the actual video parameters. The method is used to save the user's photo at the moment of authentication and to transfer the image to the server for further processing (for example, for comparison with the database). In combination with head movement analysis, this action provides complete authentication (listing 7).

Listing 7: Image comparison for final authentication.

```
.post("recog", { img: imgEncoded })
.then((response) => {
  let data = response.data;
  if (data.status == "success") {
    this.name = data.user.name;
    this.email = data.user.email;
    // this.country = data.user.country;
    this.confidence = data.confidence.toFixed(2) + "%";
    this.found = true;
    this.searching = false;
  } else {
    if (data.status == "unknown") {
      this.errorMessage = "No user was found.";
    } else {
      this.errorMessage = "No face was found.";
    }
    this.searching = false;
    this.found = false;
  }
})
.catch((err) => {
  let response = err.response;
  this.errorServer = response.status + " | " + response.statusText;
  this.searching = false;
```

During the authentication stage, a similar procedure is performed for the new user image. The constructed feature vector is compared with all available templates. The comparison is based on calculating the Euclidean distance between the vectors. If the minimum distance between the query vector and one of the user templates does not exceed the specified threshold (in most cases 0.6), the system considers the person to be recognised. A distinctive feature of this algorithm is its ability to learn. In the event of minor changes in the user's appearance (e.g., due to a change in hairstyle or facial expressions), the template can be updated with a new vector without the need to completely re-save the user profile.

The web interface, request processing mechanisms, and functions for calling external Python programs are implemented in PHP. The main scripts are save.php, which is responsible for saving new templates, and compare.php, which initiates the authentication process. The shell_exec() function is used to call Python scripts, which allows you to form a parameterised command call in the command line. This approach is simple, but with increasing load, it requires further optimisation, in particular, the implementation of asynchronous calls. The Python script main.py implements the logic of face detection, feature vector construction, reading and comparing saved templates, error handling, and

outputting the result in JSON format. This script also performs event logging. Each action (successful or with an error) is recorded in a log file with timestamps, allowing the developer or administrator to perform analysis (figure 5).

id	name	email	created_at	updated_at	biometric_data
7	Jilie	test@test.com	2025-02-26 12:52:32	2025-02-26 12:52:32	0., 8.68055562e-04, 8.68055562e-04, ...
11	Tom	tomsmit37988888@gmail.com	2025-03-18 20:46:54	2025-03-18 20:46:54	5.38194440e-02, 9.11458302e-03, 3.03819450e-03, ...

Figure 5: Logging data of the authentication process.

Vector templates are stored directly in the database (see fig. 5 (the biometric_data field)). Each template is linked to a specific user. This approach is appropriate for systems with a small or moderate number of users (up to several thousand). However, when scaling the number of records, it becomes necessary to switch to specialised systems that support vector search. These could be Faiss or Elasticsearch.

4.2. Module testing

To verify the recognition results, each authentication case was accompanied by the storage of a corresponding log. It includes a timestamp, user ID, Euclidean distance between templates, and comparison result (figure 6).

```
[2025-07-18 08:21:43] ID: 001 | Distance: 37.84 | Match: TRUE
[2025-07-18 14:32:10] ID: 002 | Distance: 66.11 | Match: FALSE
[2025-07-19 09:05:27] ID: 003 | Distance: 41.25 | Match: TRUE
[2025-07-19 18:45:03] ID: 001 | Distance: 39.77 | Match: TRUE
[2025-07-20 07:59:50] ID: 004 | Distance: 79.93 | Match: FALSE
[2025-07-20 13:22:34] ID: 002 | Distance: 44.08 | Match: TRUE
[2025-07-21 10:13:17] ID: 005 | Distance: 91.34 | Match: FALSE
[2025-07-21 16:40:52] ID: 003 | Distance: 38.56 | Match: TRUE
[2025-07-22 11:55:05] ID: 006 | Distance: 68.12 | Match: FALSE
[2025-07-23 08:08:19] ID: 001 | Distance: 36.90 | Match: TRUE
```

Figure 6: Logging of authentication results.

To objectively assess the module's effectiveness, comprehensive testing was conducted, covering functional, integration, load, and verification aspects.

Functional testing involved checking the Python module's performance with real user images. Data from open datasets (LFW and partially CelebA) was used. These datasets contain images with different lighting, backgrounds, and emotional states of the subjects [39]. As a result, the identification accuracy at a standard threshold of 0.6 was over 96.8%, confirming that the implemented algorithm meets modern requirements. The main reasons for false authentication failures were severe image darkening, face overlap, or blurring.

Integration testing revealed critical issues in data transfer between PHP and Python, in particular, sensitivity to the volume of parameters in the command line, the need to pre-check the image format, and to check for the presence of a face before starting calculations. A set of validators was developed to check for the presence of a face object before starting the encoding process. Validation of the input image before the creation of the vector representation of the face begins. It is used to check the data for relevance and validity, and prevents the calculation from crashing or reducing the accuracy of authentication. The module implements validators that check for the presence of a face in the input image before creating its numerical representation.

The validator works as follows. Before calling the face_encodings() function, the image is checked by the face_locations() function. If no face is found, the processing is terminated or an error response is returned, and only if at least one face is successfully detected, further processing is started. The method that performs validation accepts images in NumPy array format. It is first decoded from base64 or loaded via OpenCV and calls the face_recognition.face_locations() function to detect faces. If no face is found, the method returns a false value. If a face is found, it returns true and a list of coordinates of the faces found.

Load testing was performed by simulating simultaneous requests to the server using Apache JMeter tools [40]. It was found that when the load exceeded 50 requests per minute, the response time increased. This is due to the synchronous nature of shell_exec() and sequential processing of requests. Therefore, in the future, it is worth transferring the Python logic as a separate service with support for asynchronous requests using the FastAPI or Flask frameworks.

5. Conclusions

The study confirmed the need to use modern authentication methods in the context of digital transformation. Traditional methods, such as passwords and PIN codes, are not reliable enough in the context of modern cyber threats, which makes the introduction of biometric authentication methods relevant.

The paper systematised existing authentication methods and analysed the advantages of biometric approaches, particularly with regard to facial recognition. Based on this, machine learning was chosen as the basis for building an intelligent authentication module. The system architecture was developed, and initial model training was conducted, which made it possible to determine the potential of this technology in terms of improving security and usability.

Within the framework of this scientific and practical work, a full-fledged user authentication module was developed based on face recognition using OpenCV and face_recognition libraries and modern deep learning methods.

The developed module is an example of effective integration of web development tools and computer vision algorithms. Its architecture allows for biometric authentication without the use of complex or expensive solutions.

The module provides accurate user recognition in real time and is adapted to the web environment thanks to its integration with PHP. A dynamic authentication element has also been implemented, which increases the level of security by recognising micro-movements of the face.

The architecture of the authentication module is implemented as a processing pipeline, which includes the following main stages: image capture, data transfer, face detection, vectorisation, comparison with the database, decision making, and response to the client. This structure provides flexibility, scalability, and the possibility of further functionality expansion.

The research results form the basis for further practical implementation and improvement of the authentication system, taking into account the latest technological solutions. The data obtained confirm the effectiveness of the chosen approach and its practical value for implementation in various fields such as information security, education, and access control systems.

In the future, we plan to expand the functionality of the module for multi-biometric authentication, integration with mobile platforms, its adaptation for working with asynchronous API services. The module is also planned to be used in the future for user authentication in remote physical laboratories.

Author contributions

Conceptualization, Vasyl P. Oleksiuk and Yaroslav Ph. Vasylenko; methodology, Vasyl. P Oleksiuk; software, Ivan A. Hrytsai; validation, Yaroslav Ph. Vasylenko, and Ivan A. Hrytsai; formal analysis, Vasyl. P Oleksiuk, and Ivan A. Hrytsai; investigation, Ivan A. Hrytsai; resources, Ivan A. Hrytsai and Yaroslav Ph. Vasylenko; data curation, Ivan A. Hrytsai; writing—original draft preparation, Ivan A. Hrytsai; writing—review and editing, Vasyl. P Oleksiuk; visualization, Ivan A. Hrytsai; supervision, Vasyl. P Oleksiuk; project administration, Vasyl. P Oleksiuk. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Data availability statement

No new data were created or analysed during this study. Data sharing is not applicable.

Conflicts of interest

The authors declare no conflict of interest.

Acknowledgments

The study was conducted in a joint research laboratory "Digital educational innovations" of Ternopil Volodymyr Hnatiuk National Pedagogical University and the Institute for Digitalisation of Education of the National Academy of Educational Sciences of Ukraine.

Declaration on Generative AI

In the process of writing the research paper, the GPT-4o artificial intelligence model was used to search for and preliminarily systematize scientific sources on the research topic, summarize the content of individual scientific publications, and format bibliographic references. The AI model was used to translate English-language technical documentation (in particular, OpenCV) to ensure a correct understanding of the library's functionality. Claude Sonnet 4 AI was also used for preliminary analysis of the program code to identify potential syntactic and logical errors, as well as to optimize certain software solutions. GPT-4o and Gramarly were used to expand abstract and improve the English language of the paper.

References

[1] C. Katsini, M. Belk, C. Fidas, N. Avouris, G. Samaras, Security and Usability in Knowledge-based User Authentication: A Review, in: Proceedings of the 20th Pan-Hellenic Conference on Informatics, PCI '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 63. doi:10.1145/3003733.3003764.

[2] R. M. Abdullah, S. A. Hameed Alazawi, Smart Card Security Model Based on Sensitive Information, in: A. E. Hassanien, O. Castillo, S. Anand, A. Jaiswal (Eds.), International Conference on Innovative Computing and Communications, volume 537 of *Lecture Notes in Networks and Systems*, Springer Nature Singapore, Singapore, 2023, pp. 703–712. doi:10.1007/978-981-99-3010-4_56.

[3] A. K. Jain, A. Ross, S. Pankanti, Biometrics: a tool for information security, IEEE Transactions on Information Forensics and Security 1 (2006) 125–143. doi:10.1109/TIFS.2006.873653.

[4] G. G. Samatas, G. A. Papakostas, Biometrics: Going 3D, Sensors 22 (2022) 6364. doi:10.3390/s22176364.

[5] S. Albalawi, L. Alshahrani, N. Albalawi, R. Kilabi, A. Alhakamy, A Comprehensive Overview on Biometric Authentication Systems using Artificial Intelligence Techniques, International Journal of Advanced Computer Science and Applications 13 (2022). doi:10.14569/IJACSA.2022.0130491.

[6] N. Lobanchykova, T. A. Vakaliuk, V. Osadchyi, M. G. Medvediev, I. A. Pilkevych, Study of Cyber Security Approaches in Organizing Digital Voting, in: V. Sokolov, T. Radivilova, V. Ustimenko, M. Nazarkevych (Eds.), Proceedings of the Cybersecurity Providing in Information and Telecommunication Systems, CPITS 2023, co-located with International Conference on Problems of Infocommunications. Science and Technology (PICST 2023), Kyiv, Ukraine, February 28, 2023 (online), volume 3421 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 198–205. URL: https://ceur-ws.org/Vol-3421/short5.pdf.

[7] T. A. Vakaliuk, D. Talchenko, V. Osadchyi, Y. Bailiuk, O. Pokotylo, Vulnerabilities and Methods of Unauthorized Gaining Access to Video Surveillance Systems, in: V. Sokolov, T. Radivilova, V. Ustimenko, M. Nazarkevych (Eds.), Proceedings of the Cybersecurity Providing in Information and Telecommunication Systems, CPITS 2023, co-located with International Conference on Problems of Infocommunications. Science and Technology (PICST 2023), Kyiv, Ukraine, February 28, 2023 (online), volume 3421 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 174–181. URL: https://ceur-ws.org/Vol-3421/short2.pdf.

[8] I. Stylios, S. Kokolakis, O. Thanou, S. Chatzis, Behavioral biometrics & continuous user authentication on mobile devices: A survey, Information Fusion 66 (2021) 76–99. doi:10.1016/j.inffus.2020.08.021.

[9] D. S. Antoniuk, T. A. Vakaliuk, V. V. Didkivskyi, O. Vizghalov, O. V. Oliinyk, V. M. Yanchuk, Using a business simulator with elements of machine learning to develop personal finance management skills, in: A. E. Kiv, S. O. Semerikov, V. N. Soloviev, A. M. Striuk (Eds.), Proceedings of the 9th Illia O. Teplytskyi Workshop on Computer Simulation in Education (CoSinE 2021) co-located with 17th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer (ICTERI 2021), Kherson, Ukraine, October 1, 2021, volume 3083 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 59–70. URL: https://ceur-ws.org/Vol-3083/paper131.pdf.

[10] D. S. Antoniuk, T. A. Vakaliuk, V. V. Didkivskyi, O. Y. Vizghalov, Development of a simulator to determine personal financial strategies using machine learning, in: A. E. Kiv, S. O. Semerikov, V. N. Soloviev, A. M. Striuk (Eds.), Proceedings of the 4th Workshop for Young Scientists in Computer Science & Software Engineering (CS&SE@SW 2021), Virtual Event, Kryvyi Rih, Ukraine, December 18, 2021, volume 3077 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 12–26. URL: https://ceur-ws.org/Vol-3077/paper02.pdf.

[11] D. S. Antoniuk, T. A. Vakaliuk, V. V. Didkivskyi, O. Y. Vizghalov, O. V. Oliinyk, V. M. Yanchuk, Enhancing personal financial management skills through a machine learning-powered business simulator, in: S. O. Semerikov, A. M. Striuk, M. V. Marienko, O. P. Pinchuk (Eds.), Proceedings of the 7th International Workshop on Augmented Reality in Education (AREdu 2024), Kryvyi Rih, Ukraine, May 14, 2024, volume 3918 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 198–205. URL: https://ceur-ws.org/Vol-3918/paper018.pdf.

[12] Y. O. Hodlevskyi, T. A. Vakaliuk, O. V. Chyzhmotria, O. H. Chyzhmotria, O. V. Vlasenko, Finding Anomalies in the Operation of Automated Control Systems Using Machine Learning, in: T. Hovorushchenko, O. Savenko, P. T. Popov, S. Lysenko (Eds.), Proceedings of the 4th International Workshop on Intelligent Information Technologies & Systems of Information Security, Khmelnytskyi, Ukraine, March 22-24, 2023, volume 3373 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 681–698. URL: https://ceur-ws.org/Vol-3373/paper47.pdf.

[13] A. O. Poliaiev, N. N. Shapovalova, S. V. Bilashenko, A. M. Striuk, Research and development of software for hydroacoustic signal analysis using machine learning techniques, in: S. O. Semerikov,

A. M. Striuk (Eds.), Proceedings of the 7th Workshop for Young Scientists in Computer Science & Software Engineering (CS&SE@SW 2024), Virtual Event, Kryvyi Rih, Ukraine, December 27, 2024, volume 3917 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 441–450. URL: https://ceur-ws.org/Vol-3917/paper63.pdf.

[14] A. Kiv, V. Soloviev, A. Tuzhykov, T. Kavetskyy, N. Hoivanovych, O. Šauša, J. Rusnák, H. Švajdlenková, J. Ostrauskaite, A. Bielinskyi, M. Slusarenko, Machine Learning Interatomic Potentials and Orientational Defects in Polymers, in: P. Petkov, M. E. Achour, C. Popov (Eds.), Nanotechnological Advances in Environmental, Cyber and CBRN Security, NATO Science for Peace and Security Series B: Physics and Biophysics, Springer Netherlands, Dordrecht, 2025, pp. 91–104. doi:10.1007/978-94-024-2316-7_6.

[15] A. Bielinskyi, V. N. Soloviev, A. Matviychuk, H. Velykoivanenko, Interpretable Machine Learning using Visibility Graph and Random Forests, in: V. Snytyuk, L. Kirichenko, O. Mulesa, S. Lupenko (Eds.), Proceedings of the Information Technology and Implementation (IT&I) Workshop: Artificial Intelligence Technologies and Data Science (IT&I-WS: AITDS 2025), Kyiv, Ukraine, November 20 - 21, 2025, volume 4158 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2025, pp. 146–158. URL: https://ceur-ws.org/Vol-4158/Paper12.pdf.

[16] V. D. Derbentsev, V. S. Bezkorovainyi, A. V. Matviychuk, O. M. Pomazun, A. V. Hrabariev, A. M. Hostryk, A comparative study of deep learning models for sentiment analysis of social media texts, in: H. B. Danylchuk, S. O. Semerikov (Eds.), Proceedings of the Selected and Revised Papers of 10th International Conference on Monitoring, Modeling & Management of Emergent Economy (M3E2-MLPEED 2022), Virtual Event, Kryvyi Rih, Ukraine, November 17-18, 2022, volume 3465 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 168–188. URL: https://ceur-ws.org/Vol-3465/paper18.pdf.

[17] D. S. Shanurina, R. S. Savitskyi, T. A. Vakaliuk, Feature engineering and deep learning for musical interval recognition through spectral analysis, Discover Artificial Intelligence 6 (2026) 744. doi:10.1007/s44163-026-01499-3.

[18] N. Karpiuk, H. Klym, I. Vasylchychyn, Facial recognition system based on the Haar cascade classifier method, in: 2023 24th International Conference on Computational Problems of Electrical Engineering (CPEE), 2023, pp. 1–4. doi:10.1109/CPEE59623.2023.10285310.

[19] M. Dubovečak, E. Dumić, A. Bernik, Face Detection and Recognition Using Raspberry PI Computer, Tehnički glasnik 17 (2023) 346–352. doi:10.31803/tg-20220321232047.

[20] N. Boyko, O. Basystiuk, N. Shakhovska, Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library, in: 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), 2018, pp. 478–482. doi:10.1109/DSMP.2018.8478556.

[21] Y. O. Hodlevskyi, T. A. Vakaliuk, Optimal Gradient Descent Algorithm for LSTM Neural Network Learning, 2024, pp. 245–249. doi:10.1109/SIST61555.2024.10629398.

[22] A. Kumar, S. Jain, M. Kumar, Face and gait biometrics authentication system based on simplified deep neural networks, International Journal of Information Technology 15 (2023) 1005–1014. doi:10.1007/s41870-022-01087-5.

[23] V. Mukovoz, T. Vakaliuk, S. Semerikov, Road Sign Recognition Using Convolutional Neural Networks, in: E. Faure, Y. Tryus, T. Vartiainen, O. Danchenko, M. Bondarenko, C. Bazilo, G. Zaspa (Eds.), Information Technology for Education, Science, and Technics, volume 222 of *Lecture Notes on Data Engineering and Communications Technologies*, Springer Nature Switzerland, Cham, 2024, pp. 172–188. doi:10.1007/978-3-031-71804-5_12.

[24] Q. Sun, A. Redei, Knock Knock, Who's There: Facial Recognition using CNN-based Classifiers, International Journal of Advanced Computer Science and Applications 13 (2022) 9–16. doi:10.14569/ijacsa.2022.0130102.

[25] M. R. Golla, P. Sharma, Performance Evaluation of Facenet on Low Resolution Face Images, in: S. Verma, R. S. Tomar, B. K. Chaurasia, V. Singh, J. Abawajy (Eds.), Communication, Networks and Computing, volume 839 of *Communications in Computer and Information Science*, Springer Singapore, Singapore, 2019, pp. 317–325. doi:10.1007/978-981-13-2372-0_28.

[26] T. Radivilova, L. Kirichenko, V. Pantelieiev, A. Mazepa, V. Bilodid, Analysis of authentication methods for full-stack applications and implementation of a web application with an integrated authentication system, Innovative technologies and scientific solutions for industries (3 (29)) (2024) 76–90. doi:10.30837/2522-9818.2024.3.076.

[27] S. O. Semerikov, T. A. Vakaliuk, O. B. Kanevska, O. A. Ostroushko, A. O. Kolhatin, Edge intelligence unleashed: a survey on deploying large language models in resource-constrained environments, Journal of Edge Computing 4 (2025) 179–233. doi:10.55056/jec.1000.

[28] J. Deng, J. Guo, J. Yang, N. Xue, I. Kotsia, S. Zafeiriou, ArcFace: Additive Angular Margin Loss for Deep Face Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (2022) 5962–5979. doi:10.1109/TPAMI.2021.3087709.

[29] J. Deng, S. Zafeririou, ArcFace for Disguised Face Recognition, in: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 2019, pp. 485–493. doi:10.1109/ICCVW.2019.00061.

[30] A. N, K. Anusudha, Real time face recognition system based on YOLO and InsightFace, Multimedia Tools and Applications 83 (2024) 31893–31910. doi:10.1007/s11042-023-16831-7.

[31] Y. Li, J. Wang, W. Liang, H. Xue, Z. He, J. Lv, L. Zhang, CR-GAN: Automatic craniofacial reconstruction for personal identification, Pattern Recognition 124 (2022) 108400. doi:10.1016/j.patcog.2021.108400.

[32] M. Rodrigo, C. Cuevas, N. García, Comprehensive comparison between vision transformers and convolutional neural networks for face recognition tasks, Scientific Reports 14 (2024) 21392. doi:10.1038/s41598-024-72254-w.

[33] W. Su, Y. Wang, K. Li, P. Gao, Y. Qiao, Hybrid token transformer for deep face recognition, Pattern Recognition 139 (2023) 109443. doi:10.1016/j.patcog.2023.109443.

[34] S. Divya, J. T, A. V. A. Geo, Innovative Approaches to Criminal Identification Using Real Time Facial Recognition, in: 2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI), 2025, pp. 1694–1700. doi:10.1109/ICMCSI64620.2025.10883092.

[35] M. N. Saadat, H. Kabir, Z. A. Long, H. Sofian, M. F. A. Zuhairi, Efficient Face Detection And Identification In Networked Video Surveillance Systems, in: 2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM), 2020, pp. 1–9. doi:10.1109/IMCOM48794.2020.9001697.

[36] W. Sun, Y. Song, H. Zhao, Z. Jin, A Face Spoofing Detection Method Based on Domain Adaptation and Lossless Size Adaptation, IEEE Access 8 (2020) 66553–66563. doi:10.1109/ACCESS.2020.2985453.

[37] A. Swaminathan, M. Chaba, D. K. Sharma, Y. Chaba, Gender Classification using Facial Embeddings: A Novel Approach, Procedia Computer Science 167 (2020) 2634–2642. doi:10.1016/j.procs.2020.03.342, International Conference on Computational Intelligence and Data Science.

[38] O. S. Holovnia, V. P. Oleksiuk, Selecting cloud computing software for a virtual online laboratory supporting the Operating Systems course, in: A. E. Kiv, S. O. Semerikov, M. P. Shyshkina (Eds.), Proceedings of the 9th Workshop on Cloud Technologies in Education, CTE 2021, Kryvyi Rih, Ukraine, December 17, 2021, volume 3085 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 216–227. URL: https://doi.org/10.55056/cte.116. doi:10.55056/CTE.116.

[39] S. O. Semerikov, T. A. Vakaliuk, I. S. Mintii, V. A. Hamaniuk, V. N. Soloviev, O. V. Bondarenko, P. P. Nechypurenko, S. V. Shokaliuk, N. V. Moiseienko, V. R. Ruban, Mask and Emotion: Computer Vision in the Age of COVID-19, in: Digital Humanities Workshop, DHW 2021, Association for Computing Machinery, New York, NY, USA, 2022, p. 103–124. doi:10.1145/3526242.3526263.

[40] V. O. Maliarskyi, V. P. Oleksiuk, Review of modern tools for edge computing systems quality assurance, in: T. A. Vakaliuk, S. O. Semerikov (Eds.), Proceedings of the 5th Edge Computing Workshop (doors 2025), Zhytomyr, Ukraine, April 4, 2025, volume 3943 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2025, pp. 67–80. URL: https://ceur-ws.org/Vol-3943/paper16.pdf.