Pak. j. life soc. Sci. (2026), 24(1): 319-329 E-ISSN: 2221-7630;P-ISSN: 1727-4915

Pakistan Journal of Life and Social Sciences
www.pjlss.edu.pk

https://doi.org/10.57239/PJLSS-2026-24.1.0026

RESEARCH ARTICLE

Development of an Intelligent Image Processing and Face Detection System for Psycho-Emotional Monitoring

E.A. Revyakina[1]*, A.R. Gazizov[2]

[1,2]Don State Technical University, Rostov-On-Don, Russia

ARTICLE INFO

Received: Jan 21, 2026

Accepted: Apr 1, 2026

Keywords

Computer Vision
Face Detection
Image Processing
Machine Learning
Neural Networks
Autonomous Systems
Preprocessing
Psycho-Emotional Monitoring
Data Privacy

***Corresponding Author:**

ABSTRACT

This article examines the problem of increasing the robustness and accuracy of face detection in the presence of unstable input image quality, typical of real-world shooting scenarios. The relevance of the study stems from the need to create effective computer vision tools capable of operating under conditions of limited computing resources while maintaining the confidentiality of processed data and robustness to external interference. The aim of the study is to develop an intelligent image processing and face detection system for psycho-emotional monitoring, designed for autonomous local operation without the use of cloud services. The system is based on a comprehensive approach combining image preprocessing, image quality analysis, and face detection using the YOLOv8n-face model, adapted to various angles and distortions. The key scientific novelty of the study is the use of an artificial image degradation mechanism to generate a training dataset, including typical defects of real-world shooting: blur, noise, compression artifacts, and unstable lighting. A comparative study of the baseline and additionally trained models was conducted, confirming the effectiveness of the proposed approach. This paper describes a modular architecture for an application implemented in Python using Open CV, Ultra lytics, Pillow, and Tkinter libraries, ensuring flexibility and scalability. The practical significance of this work lies in the creation of a fully functional prototype of an intelligent system suitable for local use in educational, research, and corporate environments where autonomy, privacy, and resilience to external interference are critical. The results demonstrate that adapting the model to real-world conditions by training on artificially distorted data significantly improves detection reliability.

INTRODUCTION

In today's world, there is a rapid increase in the number of digital images processed in consumer, corporate, and government systems. Automatic analysis of visual content is becoming an integral part of many applications, particularly in areas such as personal identification, biometric authentication, security systems, and video surveillance. Face detection, a key component of computer vision, is widely used in psycho-emotional monitoring, banking, educational applications, access control systems, and mass-market consumer products. However, high recognition accuracy still significantly depends on the quality of the original images, shooting conditions, and the resilience of the algorithms to external interference. The human face is a complex biometric object, the recognition of which is complicated by numerous factors: head position (frontal, three-quarter, profile), uneven lighting, complex backgrounds, low resolution, digital noise, and compression artifacts (Degtyarev, Shpiryuk, 2017). Even modern mobile devices with built-in intelligent modules do not always provide stable and accurate face recognition under unfavorable shooting conditions.

The issue is particularly acute in the context of privacy and data processing autonomy. Cloud-based facial recognition services, such as the Azure Face API, Amazon Rekognition, and Google Cloud Vision API, have become widespread. While they demonstrate high accuracy, they require a constant internet connection and involve transmitting biometric data to external servers, which is not always acceptable under personal data laws and corporate security policies (Petrova, Papkova, 2024). This

is especially critical for educational, scientific, and secure environments where the transmission of sensitive information over the network is unacceptable.

Existing local solutions also have limitations. Classical methods such as Haar cascades offer high performance but are sensitive to shooting conditions and image quality (Selyutina, 2024). Histogram-Based Gradient (HOG) methods combined with support vector machines (SVM) are more robust to background and illumination, but are less effective with strong head rotations and require significant computational resources (Usmanova, 2023). Modern neural network approaches (Retina Face, Blaze Face) demonstrate impressive accuracy but often require powerful hardware, complex setup, and do not always support additional training on custom data.

This paper proposes an alternative approach: developing an intelligent system for image processing and face detection from various viewing angles, designed for autonomous local operation. The system implements comprehensive processing of visual data, including pre-filtering, image quality analysis, and robust face detection based on an optimized YOLOv8n-face model, adapted through additional training on a specially prepared dataset with artificial distortions.

The relevance of the study is determined by the need to create effective computer vision tools capable of functioning under conditions of limited computing resources, while maintaining the confidentiality of processed data and resistance to external interference. The aim of this work is to develop an intelligent image processing and face detection system that ensures high accuracy of operation under resource-limited conditions, while maintaining confidentiality and resistance to external interference.

The research's novelty lies in the comprehensive integration of image preprocessing, image quality analysis, and local face detection methods into a single, robust software system. Unlike cloud-based solutions, the proposed system requires no network connection and is completely autonomous. The use of artificial image degradation to generate the training set allows the model to adapt to typical defects in real-world footage.

The practical significance of this work lies in the creation of a fully functional prototype of an intelligent system suitable for local use under limited computing resources. The system can be applied in psychological services, educational institutions, research laboratories, enterprises, and government agencies where privacy policies are important. The research results, architecture, and program code can be used as a basis for subsequent developments and adapted for related computer vision applications, from access control to biometric verification.

THEORETICAL BASIS OF THE RESEARCH

Image Quality as a Factor in Recognition Efficiency

The effectiveness of computer vision algorithms, including face detection, directly depends on the quality of the source image. The results of visual content analysis are largely determined by the parameters of the material input to the intelligent system (Tsarev, Drzhevetsky, 2006).

Facial recognition is a complex process sensitive to numerous factors. Head position - frontal, three-quarter, or profile - can significantly distort the proportions of a face in an image, making it difficult to recognize. Lighting affects brightness and shadows, potentially obscuring important facial features such as the eyes, nose, and lips. Complex backgrounds and low contrast make it difficult to separate a face from the surrounding scene, reducing detection accuracy. Poor image quality, due to blur, digital noise, or distortion, impairs algorithm performance and can lead to errors (Rybalchenko et al., 2025).

Such issues are particularly critical in biometric systems, where the reliability of the algorithm determines not only user convenience but also security. In services that enable financial transactions using facial recognition, any error can lead to access denial or, conversely, unauthorized access. Therefore, high accuracy and resilience to external conditions are becoming mandatory requirements for such systems.

In computer vision, much attention has traditionally been paid to the quality of source data. However, modern trends are shifting this emphasis: developers are seeking to train intelligent models on more complex, low-quality photographs, increasing their robustness. Nevertheless, the

preprocessing stage remains crucial - it allows for the improvement of even weak source data, thereby increasing the stability and accuracy of algorithms (Open CV, n.d.).

Today, the data preparation stage is often underestimated. Developers primarily strive to improve the architecture of neural networks, assuming that the model should cope even with low-quality input data. However, preprocessing requires additional steps, requiring developers to modify already trained models and, in some cases, even integrate entire chains of new features, which can be labor- and resource-intensive. Initial processing before detection can be crucial: it significantly improves the quality of even defective visual data, increasing the stability and accuracy of the entire system. Thus, one of the main tasks of image processing in computer vision systems is adapting visual data to the requirements of specific recognition algorithms. Otherwise, even a powerful algorithm will not be able to provide stable and accurate results.

FACE DETECTION METHODS

Existing approaches to face detection can be roughly divided into two categories: classical methods and modern algorithms based on neural networks. Each of these approaches has its own advantages, limitations, and scope of application.

Haar cascades are one of the most widely used classical methods for face detection. The method is based on a sequence of simple filters that sequentially analyze images for characteristic human facial features. Each filter in the cascade checks a specific visual pattern - for example, the shadow of the eyes or the contour of the nose. If an image region fails to meet the criteria early on, further checking is stopped, saving resources and speeding up the process (Ultralytics, n.d.).

The histogram of oriented gradients (HOG) method, combined with a support vector machine (SVM), is another classic approach. The method essentially transforms an image into a set of features reflecting the structure and direction of contours - so-called gradients. These gradients are grouped into blocks, and their distribution is used to generate a "fingerprint" of the object. This approach allows for more accurate detection of facial shape and its boundaries, even if the image is slightly darkened or unevenly illuminated (Deys, 2022).

The HOG method is more robust to background and illumination than Haar cascades. However, it is less effective with strong head rotations, as the facial shape changes significantly and the oriented gradients become unstable. Furthermore, the method is computationally intensive, especially when processing high-resolution images, which may limit its applicability.

Modern neural network approaches demonstrate high accuracy in face detection even under challenging camera angles, low image quality, and unstable lighting conditions. Unlike traditional methods, neural network architectures are trained on large datasets and can adapt to a wide range of situations - from full-face portraits to semi-profiles and profiles (Schneider, 2018). One of the key features of such systems is the ability to detect a face not only by its general shape but also by key points (eyes, nose, mouth). This allows for more accurate determination of head position and orientation, and also increases resilience to partial occlusions and distortions.

Despite their obvious advantages, neural network approaches have a number of limitations. They require significant computing resources, especially when working with video or high-quality images. Implementing such models into applied solutions is often challenging: running on low-performance hardware can cause delays, and autonomous systems cannot function without an internet connection. It's also worth considering that in some sectors (government agencies, the financial sector), neural networks do not always meet the requirements for transparent decision-making, as their operating principles are difficult to explain in legally understandable terms.

Thus, the choice of detection method depends on the specific tasks, available resources, and system reliability and performance requirements. Classical methods offer high speed and ease of implementation, but are sensitive to image quality. Neural network models guarantee high accuracy and flexibility, but require significant computing power.

Image Preprocessing Methods

Image preprocessing is a key stage in computer vision systems. Its primary purpose is to prepare visual material for analysis, improve image quality, and eliminate defects that could negatively

impact the algorithms' performance. This stage is especially important when working with real-world photographs taken under unstable conditions, such as poor lighting, digital noise, or complex backgrounds (Kravchenko et al., 2020).

Image processing methods can be roughly divided into two main groups based on the detection approach chosen: black-and-white processing and color processing. Each group can employ various auxiliary processing techniques to improve the efficiency of subsequent analysis. Sharpening is often used regardless of the color space, especially when working with blurry or low-contrast images. This process allows for better highlighting of key facial details - the contours of the eyes, lips, and nose - which facilitates more accurate recognition.

Distortion removal and noise reduction are important pre-processing steps. Distortions can arise from poor-quality photography, camera movement, or poor lighting. Modern digital filtering methods can reduce the visibility of such defects and, in some cases, eliminate them completely. This is especially important when working with images captured in real-world, less-than-ideal conditions.

Grayscale conversion is used when processing in monochrome. This approach reduces the impact of color variations in skin, clothing, and the background, reduces the amount of data to analyze, and improves algorithm performance. Working in monochrome also simplifies the process of extracting edges and key features, as the system focuses on brightness distribution rather than color composition.

Color segmentation is used in color image processing. The method involves dividing the image into regions with uniform color characteristics. This allows for the identification of potentially important areas and the elimination of background, reducing the workload of subsequent processing steps. One particular example of this segmentation is skin tone detection, based on the assumption that human skin has characteristic color ranges described by specific color spaces. Image preprocessing allows you to adapt visual material to algorithm requirements, improve its quality, remove distortions, and identify key features. The use of methods such as grayscale conversion, sharpening, noise filtering, and color segmentation significantly improves recognition accuracy and stability, especially when working with real, imperfect visual data.

Existing Technologies and Tools

In the field of computer vision, there is a wide range of technologies, libraries, and platforms designed for image processing and facial recognition. These tools vary significantly in their operating principles, computational resource requirements, accuracy, and ease of implementation.

Open CV is one of the most popular and versatile computer vision libraries. It offers a wide range of functions for working with images and videos, from simple transformations (filters, grayscale conversion, histogram equalization) to more complex operations (object detection, face recognition, object tracking). Open CV implements both classic Haar cascades and integrates with modern neural network models, making it suitable for both simple and advanced projects (Korochent sev et al., 2021; Redmon et al., 2016).

Dlib is actively used for tasks related to key point analysis and biometric model construction. The library implements face detection based on the histogram of oriented gradients (HOG) method in combination with support vector machines (SVM), and also offers ready-made models for detecting 68 facial keypoints. However, Dlib is more resource-intensive and is not always suitable for real-time processing on low-end hardware.

More modern solutions include Google's MediaPipe and RetinaFace, which are based on deep neural networks. These tools demonstrate high accuracy even in challenging conditions (head rotation, low image quality, partial occlusion). However, their use can be hampered by the need for pre-training the neural network, connection to a GPU, or high RAM requirements (Deng et al., 2020; Kodatsky et al., 2024).

There are also cloud-based solutions, such as the Azure Face API (Microsoft), Amazon Rekognition, and Google Cloud Vision API. They enable facial detection and analysis without requiring local implementation of algorithms. These systems offer high accuracy and speed, but they require a

constant internet connection and are not always suitable for working with personal or biometric data due to privacy and security concerns.

YOLOv8 (You Only Look Once) by Ultra lytics is a modern neural network architecture for object detection, including a specialized version, YOLOv8n-face, for face detection. The model demonstrates good accuracy with minimal computational requirements, making it particularly suitable for on-premises use. A key advantage is the ability to further train on custom datasets, allowing the model to be tailored to specific operational conditions.

A comparative analysis of existing solutions shows that commercial cloud services provide high accuracy but are unsuitable for standalone applications and environments with high privacy requirements. Classic libraries such as Open CV and Dlib are well suited for local work but are inferior to neural network approaches in accuracy at complex angles. Modern neural network models require significant resources and are difficult to configure.

The system developed in this study proposes an alternative approach focused on autonomous local image processing. It is based on an optimized YOLOv8n-face model, adapted through additional training on a specially prepared dataset with artificial distortions. This architecture enables high adaptability, stability in offline mode, and flexibility for use in resource-constrained environments.

Software Structure

General Algorithm of the Program

The intelligent face detection system operates according to a strictly defined scenario, implementing the entire image processing lifecycle - from loading source material to displaying and saving results. The algorithm is built on the principles of modularity and separation of responsibilities between components, ensuring ease of maintenance and scalability.

Each stage of the algorithm performs an independent task, yet is closely linked to the previous step. This approach ensures predictability of the program's operation and allows for easy expansion of the system - for example, by adding new preprocessing methods or integrating alternative recognition models.

The user can select one of two available detection models: the basic YOLOv8n-face or the YOLOv8n-face model trained on distorted data. The choice is made via radio buttons in the graphical interface. When selecting a model, the system automatically switches to the corresponding weights and configuration, allowing for a correct comparison of results between the standard and adapted models. This mechanism ensures experiment flexibility and makes it possible to evaluate the impact of additional training on detection quality under various shooting conditions.

Choosing a Programming Language

The Python programming language was chosen as the basis for developing the intelligent system. The main factors influencing this choice were its rapid development, rich library ecosystem, and cross-platform compatibility.

Python has a clear and easy-to-read syntax. The lack of explicit variable type declarations and a uniform indentation system allow for quick code writing and modification. This allows developers to quickly test ideas, create program prototypes, and make edits without wasting time on complex code structures. Python has a vast ecosystem of libraries that significantly simplify a variety of tasks, including image processing, developing machine learning models, and creating graphical interfaces.

Python runs equally well on Windows, Linux, and macOS. Code written on one platform can be run on another without significant modification. Its lack of OS dependency allows for distribution to a wide range of users, from students to professionals working on Linux servers.

The Open CV library provides a full range of functions for working with images and video, from file reading to complex filtering and geometric transformations. Ultra lytics YOLOv8 facilitates the integration of modern neural networks for object detection, allowing for quick loading of pre-trained models and detection. The Pillow library serves as a reliable tool for loading, converting, and saving images in various formats (JPEG, PNG). The built-in Tkinter module allows for the creation of full-

fledged window interfaces without additional dependencies, offering a wide range of widgets for displaying graphics and managing user input.

When choosing a programming language, alternative options were also considered. C++ provides the highest execution speed, but requires extensive code and complex configuration of third-party libraries. Java is inferior to Python in terms of ease of use with computer vision libraries, and connecting Open CV or Tensor Flow requires additional linking mechanisms, which complicate development. C# for .NET offers a sophisticated platform for Windows, but extending to other operating systems requires the use of additional technologies. MATLAB offers powerful built-in analysis tools, but its commercial license makes it unattractive for academic research on a limited budget.

An additional factor in favor of Python was the ease of organizing a complete machine learning infrastructure - from dataset preparation to model training and integration. Thanks to the Ultralytics module, development and subsequent training of the YOLOv8 model with custom annotations and key points took place within a single ecosystem, without the need to write and adapt code for external libraries.

Thus, Python demonstrates the best combination of development speed, availability of necessary tools, and deployment versatility, which led to its choice as the main language for implementing all modules of the intelligent system.

Libraries and Tools Used

The intelligent image processing and face detection system was implemented using libraries and tools that met the project's goals and ensured stable operation when running locally. When selecting components, primary priority was given to open-source, freely distributed solutions with broad community support and good documentation. This approach reduced development time, simplified integration, and ensured flexibility for future functionality expansion (Bazarevsky et al., 2019; King, 2009).

Open CV has become a core component for computer vision tasks. The library has been used for image preprocessing (filtering, contrast enhancement, noise reduction, edge detection), as well as for a number of auxiliary operations, such as color space conversion and scaling.

Face detection is implemented using the YOLOv8n-face model provided by the Ultra lytics library. This model is a lightweight version of the YOLO object detection algorithm, specifically adapted for working with faces. The neural network demonstrates good accuracy with minimal computational requirements, making it particularly suitable for local use. A key feature is support for facial key point detection, which expands the analysis capabilities and allows for more accurate head orientation determination.

During testing of the basic YOLOv8n-face model, it was found that image preprocessing can have both positive and negative effects on recognition accuracy. In some cases, filtering improved visual quality and increased face recognition results, but in some photos with complex distortions or low quality, the effect was the opposite: the model either failed to detect faces or their localization accuracy decreased.

This observation necessitated further training of the model. To prepare the training dataset, a procedure for artificial image degradation was developed using the Num Py, Open CV, Random, and glob libraries. A chain of functions was implemented generating various distortions: Gaussian blur (simulating poor focus), salt-pepper noise (digital artifacts), darkening/brightening (unstable lighting), and low-quality JPEG compression (compression artifacts). The transformations were applied both individually and in random combinations with varying intensities, allowing for the creation of a diverse training set that enhances the generalization ability of the model.

The Pillow library was used to convert images to a format compatible with the GUI, scale them, and display them in a user-defined window. Thanks to its support for various formats and ease of use, Pillow became a convenient solution for visualizing processing results.

The graphical interface is based on the Tkinter module, which is included in the standard Python distribution. Tkinter allows you to create windowed applications with buttons, text fields, selection

controls, and image displays. The module ensures minimal system load and does not require the installation of additional components, which is especially important when developing cross-platform standalone applications (Dzhurov et al., 2023).

The combined toolset supports the full image processing cycle: from data collection and degradation to model training and output in a graphical interface. The combination of OpenCV, Ultralytics, Pillow, and Tkinter demonstrates high flexibility and enables both scientific experiments and applied solutions without dependence on cloud services.

Software Design

General System Architecture

The application's architecture is built on a multi-layered principle and includes four key layers, each responsible for a distinct set of functions and interacting with adjacent layers through clearly defined interfaces (Figure 1). All four layers are interconnected and operate sequentially. Each layer begins its work only after the previous one has completed. This algorithm ensures predictable results regardless of the number of times the program is used.

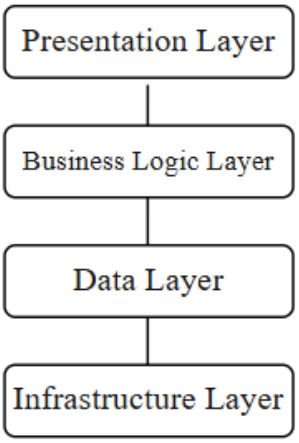

Figure 1: Multilayer application architecture

The presentation layer is responsible for user interaction and displaying the results of the system's work. It is through this layer that the first impression of the program is formed. A clear, responsive, and intuitive application window inspires confidence. The ease of use of controls (buttons, tabs, dialogs) directly impacts the speed and quality of task execution. This layer is based on the Tkinter library. Upon startup, a main window is created containing controls: radio buttons for selecting between the base and trained models, a button to start processing, a button to save the results, and a log field for displaying messages.

The business logic layer is responsible for implementing the application's core functions - those that directly process images, make decisions about filter application, and run the face detection algorithm. Unlike the presentation layer, which handles visual interaction, the business logic focuses on what exactly needs to be done and in what sequence to achieve the correct result. This layer analyzes input data and makes key decisions. Brightness and sharpness assessment methods transform abstract images into numerical characteristics. Based on these parameters, decisions are made about whether to apply specific filters.

At the business logic level, integration with YOLOv8 neural models is implemented. The detection module is abstracted from the specific implementation. The program is designed so that whether the base or additionally trained model is used, the interface and methods remain consistent. This allows experimentation with different weights and architectures without changing the rest of the code. Furthermore, this layer handles frame rendering and preparation of the final image for display. This is where the core functionality of the system is built - from converting a pixel array into user-friendly rectangles to recording a detailed log of each processing step.

The data layer is the foundation of the application, providing reliable and unified storage, access, and management of all types of information required by the system. All input images used to train the model, as well as those submitted by the user for processing, are stored in clearly structured folders.

This organization allows for automated data loading into the preprocessing and training modules, eliminating manual searches and preventing errors when specifying paths.

The catalog stores both pre-trained base models and those additionally trained on a custom dataset. The data layer is responsible for checking their presence at application startup and ensures their correct loading into memory for subsequent use. The user doesn't need to manually search for or move files - simply configure the catalog. The data layer stores all settings in configuration files, ensuring repeatable experiments and making resource management transparent. When new data appears or the folder structure changes, it's enough to edit the configuration rather than the main code, significantly simplifying maintenance.

The infrastructure layer ensures the smooth and predictable operation of all other components. It handles basic functions such as logging, file system management, support, and exception handling. This layer allows all other modules to focus on their application tasks without worrying about the intricacies of running in different environments. This allows for centralized collection of detailed processing information without duplicating code in business logic or presentation.

The program's minimalist interface allows the user to quickly navigate its functions. The program features two panels (before and after processing) for visual comparison of applied filters, implemented using widgets and the Pillow library. Images are converted from Open CV to a format suitable for display in Tkinter and displayed side by side, allowing for immediate evaluation of the preprocessing's effectiveness.

Along with real-time visualization, text messages are displayed in the logging field. All algorithm actions, statistics on applied filters, the number of faces found, and any errors are displayed directly in the application window. If necessary, the entire log history can be opened in a separate full-screen window, facilitating the analysis of lengthy processing sessions.

Taken together, this architecture ensures that the application remains flexible, extensible, and reliable: the user receives a user-friendly interface, the business logic has a clear and predictable sequence of actions, data is stored in an organized manner, and the infrastructure ensures the development and maintenance of the system.

Selecting and Training a Machine Learning Model

When building an intelligent system, a crucial step was determining the optimal machine learning model. The key selection criteria were: high accuracy when working with images in unstable conditions, processing speed, the ability to run autonomously without an internet connection, and the ability to further train the model on its own data.

YOLOv8n-face, a lightweight version of the YOLOv8 neural network architecture specifically adapted for face detection, was chosen as the main model. This model proved to be the optimal solution due to several advantages.

The model boasts high processing speed, enabling detection even on devices with limited computing power, including laptops and personal computers without a graphics accelerator. Its compact architecture requires minimal memory, which is especially important when developing offline applications. A key advantage of YOLOv8n-face is its support for facial key point detection - the model can detect not only a rectangular area but also five characteristic landmarks: the left and right eyes, the nose, and the corners of the mouth (Cherckesova et al., 2025; Vorontsov, Galanina, 2025). This expands the analysis capabilities, enabling more accurate determination of head orientation and facial expression. Furthermore, the model easily adapts to user datasets thanks to the built-in Ultra lytics library API, significantly simplifying the process of additional training and customization.

Before switching to YOLOv8n-face, the project initially used a cascade model based on the Haar algorithm included in Open CV. This method proved itself as a fast and simple method for face detection and was convenient during the initial development stage. However, significant limitations were identified during testing. The cascade model does not support additional training, making it impossible to adapt to specific conditions. For more stable operation, it was necessary to use multiple XML files simultaneously (frontal, profile, and eyes), which increased the complexity of setup and reduced the system's versatility. Non-standard camera angles often required manual adjustment of the order of model application and their parameters.

Alternative options include Retina Face, Blaze Face, or a Dlib-based solution with 68-keypoint regression. However, Retina Face requires more powerful hardware and complex setup. Blaze Face does not support additional training and is unstable with low-quality images. Dlib only performs well with frontal facial orientations, losing accuracy with head rotation.

Taking all factors into account, YOLOv8n-face demonstrated the best balance between performance, recognition accuracy, and training flexibility, which led to its selection for this study. To improve the model's resilience to distortion and unusual situations, an expanded training set of images was created, including both high-quality photographs and images with artificial defects simulating real-world shooting conditions.

Using a custom-developed script integrating Open CV, Num Py, and Random libraries, various transformations were applied to the source images: Gaussian blur, salt-pepper noise, random noise, brightness and contrast adjustments, and low-quality JPEG compression. The transformations were applied both individually and in random combinations with varying intensities.

During model training, the annotation structure was described in a YAML configuration file, which specified the paths to the folders containing the training and validation images. The file included a description of the categories (only one, "face," was used in the project), a list of facial key points (left and right eyes, nose, left and right corners of the mouth), and their relationships in the form of a skeleton for visualization.

Training was conducted over 50 epochs, ensuring gradual model adaptation without the risk of overfitting. The input image size was 640x640 pixels, and the batch size was 16 images. Training results (model weights, logs, and visualizations) were saved in a separate project folder. Training was performed locally, and thanks to its compact size and optimized architecture, YOLOv8n proved feasible even under limited computing resources.

Training was initiated using the Ultra lytics library. A model object was created based on the pre-trained yolov8n-face.pt weights, after which the training process was launched using the train () method using the data_pose. yaml configuration file.

At each training stage, loss values, accuracy metrics (precision, recall), and mAP (mean average precision), calculated for both rectangles and key points, were written to the console and log file. Images with overlaid model predictions were also generated for visual inspection. After training was completed, a manual accuracy check was performed on a new set of images not used in the training process. This check confirmed that the newly trained model performed reliably even with low-quality images, including head rotations, low lighting, and noise.

Comparative testing of the baseline and trained models on complex images revealed a significant advantage for the adapted version. The baseline model often failed to cope with even the least complex cases of defective photographs.

Therefore, to ensure correct preprocessing and more accurate recognition, the model must be trained for specific operating conditions. Thanks to its built-in learning mechanism, YOLOv8n-face enabled adaptation to new data without requiring changes to the user interface or core application logic. The model was easily integrated into a local application, fully functional without internet access.

Client Application and User Interface

When the program starts, a graphical interface is built using the Tkinter library. A main window is created with key controls: radio buttons for selecting a model, a button to start the analysis, a button to save the results, and a log field. At the same time, the necessary libraries are initialized and the face detection models (base and trained) are loaded. Loading is performed once at the start of the program, which reduces delays during subsequent launches.

After clicking the start processing button, an image selection dialog box opens. The user selects the desired file from disk. Before proceeding, the system checks the path validity by checking for Cyrillic characters in the file name or path. This check is necessary because Cyrillic characters can cause errors when running some libraries. If the path is invalid, the user receives a warning via a modal

window, and a corresponding message is displayed in the logs. If the check is successful, the image is loaded in BGR format using Open CV.

After loading, the image is transferred to the automatic analysis unit, where two key quality parameters are calculated: illuminance (the average luminance value in the HSV model) and sharpness (the variance of the Laplace operator). A full set of preprocessing filters is then applied, including 10% upscaling, noise reduction, CLAHE contrast equalization, and bilinear filtering.

The filtered image is passed to the selected YOLOv8n-face model. The detection result is a structure containing the coordinates of detected faces and key points. If no faces are found, the user receives a notification. The next step is to draw the results over the processed image: green rectangles around each detected face. The resulting image is displayed on the right side of the interface next to the original.

The user can click on the final image to open it in full-screen mode. This feature is especially useful when working on a small monitor or when presenting results to a large audience. The final step is the ability to save the processed image. Clicking the "Save Result" button opens a path selection dialog box. The system additionally checks the path for Cyrillic characters, warning the user if necessary.

During program execution, all events and actions occurring within the system are recorded (logged) and displayed in a dedicated text area located at the bottom of the program window. The log displays real-time information about the loaded image, applied filters, model parameters, number of faces found, and possible errors. If necessary, the log can be opened in a separate full-screen window.

The developed intelligent system, compared to cloud services, operates entirely locally, without an internet connection, ensuring the privacy and security of biometric data. This allows the system to be used in educational, research, and corporate environments with increased information security requirements.

CONCLUSION

The developed intelligent system for image processing and face detection from various angles is a complete software solution demonstrating high performance in local, autonomous face recognition tasks. Research has confirmed that additional training of the YOLOv8n-face model on a dataset with artificial distortions significantly improves detection resilience to typical defects of real-world footage: blur, noise, unstable lighting, and compression artifacts.

The key result of the study is experimental confirmation of the effectiveness of additional model training on data with artificial distortions. A comparison of the baseline and adapted versions of YOLOv8n-face demonstrated a significant advantage of the trained model when processing images with typical defects: blur, noise, unstable lighting, and compression artifacts. This demonstrates the importance of considering real-world operating conditions during the training data preparation stage.

A significant advantage of the development is its adherence to sustainable development principles: the system is optimized for energy-efficient operation, requires no additional hardware, does not create redundant file copies, and can be used in isolated offline environments. Local data processing ensures complete confidentiality of biometric information, which is critical for use in educational, research, and corporate environments with increased data security requirements.

The application of the developed system in the field of psycho-emotional monitoring opens up the following promising areas: early detection of stressful conditions, support for individuals with post-traumatic stress disorder (PTSD), adaptive education and an inclusive environment, predictive analytics of the psycho-emotional state of students (including participants in military conflicts) in the university educational environment.

REFERENCES

Bazarevsky V, Kartynnik Y, Vakunov A, Raveendran K, Grundmann M. BlazeFace: Sub-millisecond neural face detection on mobile GPUs. arXiv:1907.05047 [cs.CV], 2019. https://doi.org/10.48550/arXiv.1907.05047

Cherckesova L, Revyakina E, Lyashenko N. Postquantum merkle signature based on modified lampport algorithm. J Theor Appl Inf Technol 2025,103:4594-606.

Degtyarev YuS, Shpiryuk KS. Review of the current state of the sphere of computer vision. Ekonomika i sotsium 2017,10:633-8.

Deng J, Guo J, Zhou Y, Ververas I, Kotsia I, Zafeiriou S. RetinaFace: Single-shot multi-level face localization in the wild. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 13-19, 2020, Seattle, WA, USA. New York: IEEE; 2020. p. 5202-11. https://doi.org/10.1109/CVPR42600.2020.00525

Deys VI. Komp'yuternoye zreniye [Computer vision]. In: Sapprykin AA (Ed.), Progressivnyye Tekhnologii i Ekonomika v Mashinostroyenii [Progressive Technologies and Economics in Mechanical Engineering]: Proceedings of the 13th All-Russian Scientific and Practical Conference for Students and Young People. Tomsk: National Research Tomsk Polytechnic University; 2022. p. 120-1.

Dzhurov AA, Cherkesova LV, Revyakina EA. Software tool for detecting fake video content using the Deepfake technology of the GAN algorithm. H&ES Res 2023,15:60-7. https://doi.org/10.36724/2409-5419-2023-15-4-60-67

King DE. Dlib-ml: A machine learning toolkit. J Mach Learn Res 2009,10:1755-8.

Kodatsky NM, Revyakina EA, Gazizov AR. System analysis and information processing to solve the problem of detecting breakdowns of computer information storage. Herald Dagestan State Tech Univ. Tech Sci 2024,51:87-98. https://doi.org/10.21822/2073-6185-2024-51-4-87-98

Korochentsev DA, Cherckesova LV, Revyakina EA, Bordyrikhin NV, Safaryan OA. Importozameshchayushchiye Tekhnologii Obespecheniya Informatsionnoy Bezopasnosti i Zashchity Dannykh [Import-Substituting Technologies for Information Security and Data Protection]. Rostov-on-Don: Don State Technical University; 2021.

Kravchenko SV, Alekseev AV, Orlova YuA, Grinin IL, Matyushechkin DS. Problems of detecting an object in an image in deep learning problems in the field of computer vision based on convolutional neural networks. Innov Invest 2020,6:194-7.

Open CV. OpenCV documentation. n.d. https://docs.opencv.org/4.x/

Petrova DA, Papkova VA. Biometric personal data in the light of new legislation. Legal Policy and Legal life 2024,3:329-36. https://doi.org/10.24412/1608-8794-2024-3-329-336

Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 27-30, 2016, Las Vegas, NV, USA. New York: IEEE; 2016. p. 779-88. https://doi.org/10.1109/CVPR.2016.91

Rybalchenko AM, Revyakina EA, Gazizov AR. Adaptive intelligent system for pre-processing images and detecting faces at various angles of view. Nanotekhnologii: nauka i proizvodstvo 2025,4:54-8.

Schneider AS. Detektirovaniye ob"yektov na izobrazheniyakh na primere modeli YOLO [Detection of objects in images using the YOLO model as an example]. Trudy molodykh uchenykh Altayskogo gosudarstvennogo universiteta 2018,15:358-61.

Selyutina OG. Main directions of digital modernization of the banking sector in Russia. Educ Sci Sci Person 2024,3:188-90. https://doi.org/10.24412/2073-3305-2024-3-188-190

Tsarev AG, Drzhevetsky AL. Metod predvaritel'noy obrabotki graficheskikh izobrazheniy [Method of preliminary processing of graphic images]. Trans Int Symp Reliab Qual 2006,2:291-2.

Ultralytics. Ultralytics YOLOv8 Documentation. n.d. https://docs.ultralytics.com/

Usmanova NF. Analysis of graphic image processing algorithms. Sci Bull 2023,5:625-8.

Vorontsov KE, Galanina NA. Sozdaniye algoritma vektornoy bazy raspoznavaniya lits lyudey svertochnoy neyronnoy seti [Development of a vector base algorithm for facial recognition using a convolutional neural network]. In: Voitov IV (Ed.), Informatsionnyye Tekhnologii. Fizika i Matematika [Information Technologies. Physics and Mathematics]: Proceedings of the 89th Scientific and Technical Conference of Faculty, Researchers, and Postgraduate Students (with International Participation), February 3-18, 2025, Minsk, Belarus. Minsk: BSTU; 2025. p. 195-8.