

Face Recognition System Using CLIP and FAISS for Scalable and Real-Time Identification

You

May 6, 2025

Abstract

Face recognition is increasingly being adopted in industries such as education, security, and personalized services. This research introduces a face recognition system that leverages the embedding capabilities of the CLIP model. CLIP, a model trained on multimodal data, such as images and videos, generates high-dimensional features, which are then stored in a vector store for further queries. The system is designed to facilitate accurate real-time identification, with potential applications in areas such as attendance tracking and security screening. Specific use cases include event check-ins, implementation of advanced security systems, and more. The process involves encoding known faces into high-dimensional vectors, indexing them using a vector store FAISS, and comparing them to unknown images based on cosine similarity. Experimental results demonstrate a high accuracy that exceeds 90% and a performance efficiency even in datasets with a high volume of entries.

Introduction

In today's digital age, facial recognition technologies play a key role in transforming how we interact with computer systems, secure spaces, and optimize everyday processes. Facial recognition is one of the fastest growing fields in biometric authentication, enabling fast and reliable identification of individuals without the need for physical contact or additional interaction[1]. With the increasing availability of advanced computational resources and sophisticated algorithms, implementing facial recognition systems has become an indispensable tool across various industries.

The applications of this technology are diverse and span multiple sectors. In education, facial recognition can be used for automated student attendance tracking, allowing for more efficient monitoring and record-keeping. In the security sector, this technology enables the identification of suspicious individuals and enhances surveillance in public and private spaces, significantly contributing to crime prevention and increased safety. Additionally, in personalized services, facial recognition helps tailor user experiences by providing access to relevant information and services based on the user's identity[2].

The goal of this research is to develop and evaluate an efficient facial recognition system that leverages the capabilities of the CLIP (Contrastive Language-Image Pre-Training) model to extract high-dimensional features from images and video recordings. This system, based on vector search techniques used in FAISS (Facebook AI Similarity Search), enables fast and accurate real-time facial identification. Through experimental evaluation, the research aims to demonstrate how the combination of advanced machine learning techniques and optimized search algorithms can enhance the accuracy and performance of existing facial recognition systems, with potential applications in security systems, attendance tracking, and other relevant fields.

Related Work

FAISS (Facebook AI Similarity Search)[3] and CLIP (Contrastive Language-Image Pretraining)[4] represent two advanced approaches in the fields of computer vision and natural language processing for data processing and retrieval. FAISS is a highly efficient system for vector similarity search, optimized to handle large databases. Its applications include image retrieval based on visual similarity, biometric facial recognition systems, recommendation systems, and semantic search for textual data and genetic

sequences. It is particularly useful in IoT data analysis and anomaly detection, enabling fast comparisons of time series data and sensor patterns. Furthermore, FAISS is utilized in accelerated document search systems, user behavior analysis, and real-time data classification, significantly improving the scalability and efficiency of machine learning models.

On the other hand, CLIP is a transformer-based model trained on a large dataset combining images and descriptive texts, allowing it to understand visual content in the context of natural language. Its applications include image retrieval based on textual queries, automatic image captioning, zero-shot classification, and improving generative image models such as DALL-E and Stable Diffusion. CLIP is also used for content moderation, video analysis, autonomous system interactions with the environment, and in creative industries for generating artistic visuals based on semantic descriptions.

Although FAISS and CLIP have been applied individually in various domains, no implementation has been recorded that combines them for scalable real-time facial recognition. This work introduces the first such approach, leveraging CLIP’s capabilities to generate robust facial representations and FAISS’s efficiency for extremely fast searches within large databases. The integration of these two models enables the development of advanced multimodal data processing systems, providing more efficient search and classification methods across a wide range of applications, from security systems to creative industries. Our implementation not only enhances the speed and accuracy of facial identification, but also unlocks new possibilities in the field of computer vision that have not yet been explored.

Methodology and implementation

The described system is implemented in Python using CLIP for embedding extraction from images and FAISS for fast search. Video recordings of human faces are processed into frames. The dataset consists of 50 classes, each class representing one individual, with two videos per class — one for ”training” and the other for ”validation.”

For facial recognition, the system first loads known face images using the `load_known_faces` function, which iterates through directories labeled with class names (individuals). Each class contains a subdirectory named `train`, where images used for ”learning” are stored. For each image in the training directory, the function `add_known_face()` is called, which extracts the image embedding using the CLIP model and associates it with the corresponding class. This process builds a database of known faces, that is later used for recognition.

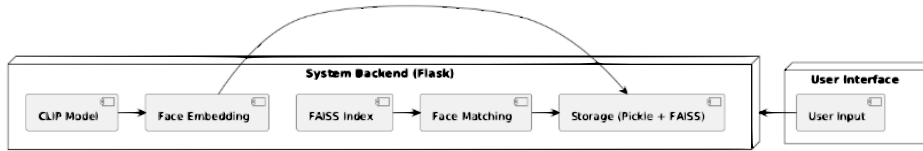


Image 1: Schema

overview.

After extracting embeddings, the computed vectors are organized into a FAISS index using the `build_index` function. FAISS enables fast vector search in high-dimensional space using the L2 distance, also known as Euclidean distance.[5]

The L2 distance between two vectors $A = (a_1, a_2, \dots, a_n)$ and $B = (b_1, b_2, \dots, b_n)$ is defined as:

$$d(A, B) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$$

Image 2: L2 distance.

This metric measures the physical distance between two points in an n-dimensional space and is used to compare face embeddings (elements of each face) — the smaller the distance, the more similar embeddings. FAISS enables efficient nearest-neighbor search using this metric, speeding up the recognition process and avoiding linear search.

In our system, FAISS is used by storing all known face embeddings in an `IndexFlatL2` index, which allows for fast search based on L2 distance. When adding a new embedding, it is simply inserted into the index along with its corresponding class label. When recognizing a new face, the embedding

of the input image is compared with the embeddings in the FAISS index, and the system finds the k2 nearest neighbors (embeddings with the smallest L2 distance). These results are then used for classification through a voting method using the `classify_face()` function among the first k1 results, the class that appears most frequently is selected by counting all occurrences of each class in that set. If the similarity threshold is not met, additional results (k1 to k2) are included to improve the decision.

The class with the most votes becomes the final prediction, unless no class meets the criteria, in which case the face is classified as Unknown.

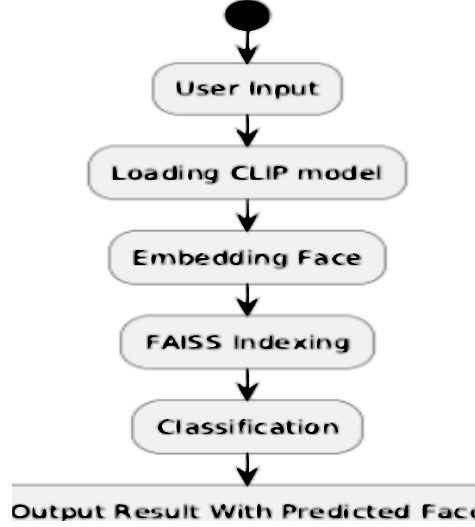


Image 3: System flowchart.

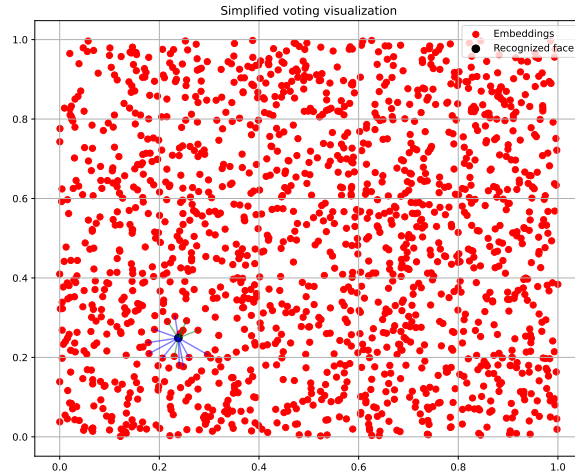


Image 4: Simplified voting

visualization.

Experiments, tests and results

First, the system was tested on a dataset of approximately 500 classes, where each person had 1 video, using some frames for training and the rest for validation (80-20 split). In this case, the results showed 100% accuracy because the training and validation data were nearly identical. Furthermore, to achieve more realistic results, the ytfaces[6] dataset was used, which consists of videos of people in different environments. This approach was chosen because human faces in videos can successfully simulate the input the system would receive when using camera input for live recognition. From the dataset, 2 videos were extracted for each class to simulate facial scanning under different conditions during evaluation.



Image 5: An example of train image.



Image 6: An example of val image.

Evaluation process

After defining the dataset, an extensive procedure was carried out to test various combinations of parameters k_1 , k_2 , and threshold. A grid search[7] was used for an efficient parameter search. The following parameter ranges were defined: k_1 : 1-9 k_2 : 2-10 threshold: 0.5 - 1.0 with a step of 0.01. Additionally, a condition was defined to skip combinations where k_1 and k_2 are equal and where k_2 is smaller than k_1 , in order to avoid meaningless steps in the search. The testing and evaluation process was carried out by forwarding each image from all validation folders to CLIP to create its embeddings. Then, the embeddings were normalized for easier comparison using `linalg.norm`[8].

For each embedding from the validation folder, a class prediction, i.e., face recognition, was performed. The true label and predicted label values were also saved for each image, which defines the actual class to which the image belongs and the class predicted by the model. If they are the same, it is a correct classification. If they differ, it indicates an error in prediction. The prediction is made by taking each validation embedding and comparing it with the known embeddings extracted from the training images, finding the most similar embedding (with the smallest distance). Once the system determines which class the most similar embeddings belong to, the validation embedding is assigned to that same class and recorded as a prediction, with an automatic accuracy check.

The minimum threshold was intentionally set to 0.5 because predictions with a lower threshold would be meaningless due to excessive strictness. The higher the threshold, the more images are considered, and the number of predictions increases. (Lower threshold... stricter decision and vice versa.)

During the evaluation of the results, the values obtained for each step of the search were automatically saved in the file `grid_search_results`. Specifically, it contains the following values: k_1 , k_2 , threshold, precision, recall, f1, tp, fp, tn, fn.[9]

tp	True positive predictions, i.e., correctly predicted positive samples
fp	False positive predictions, i.e., incorrectly predicted positive samples
tn	True negative predictions, i.e., correctly predicted negative samples
fn	False negative predictions, i.e., incorrectly predicted negative samples
Precision	How many of all predicted positive samples are actually positive
Recall	How many positive samples are correctly predicted as positive
F1-score	The harmonic mean of precision and recall (their balance)

Table 1: Explanation of classification metrics

As the output of each step of the grid search, a row of data was generated with the k1, k2, and threshold parameters, along with the results produced by that combination of parameters. For example: k1, k2, threshold, precision, recall, f1, tp, fp, tn, fn (1, 2, 1.0, 0.7305, 0.7305, 0.677, 2702, 991, 46577, 991) Each individual parameter combination generated evaluation metrics that allowed for the analysis of the model’s performance. The results were displayed in tabular form, including the used parameter values and the obtained metrics: precision, recall, F1-score, as well as the absolute values of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). This output provides a detailed understanding of how changes in hyperparameters affect classification performance, enabling the selection of the optimal combination that achieves the best balance between the model’s output metrics.

For example, the configuration k1=1, k2=2, and threshold=1.0 resulted in precision=0.7305 and recall=0.7305, with the corresponding number of correct and incorrect classifications (TP=2702, FP=991, TN=46577, FN=991). By analyzing numerous results of this type, it is possible to assess the ideal balance between different metrics and adjust the model to achieve optimal classification efficiency in face recognition.

During the evaluation, the system’s performance was monitored for different dataset sizes. As one might logically conclude, the metrics mainly decrease as the number of classes and images increases. Below 25 classes, the performance was perfect because the number of embeddings was small enough to prevent misclassifications due to the ample space in the embedding space.

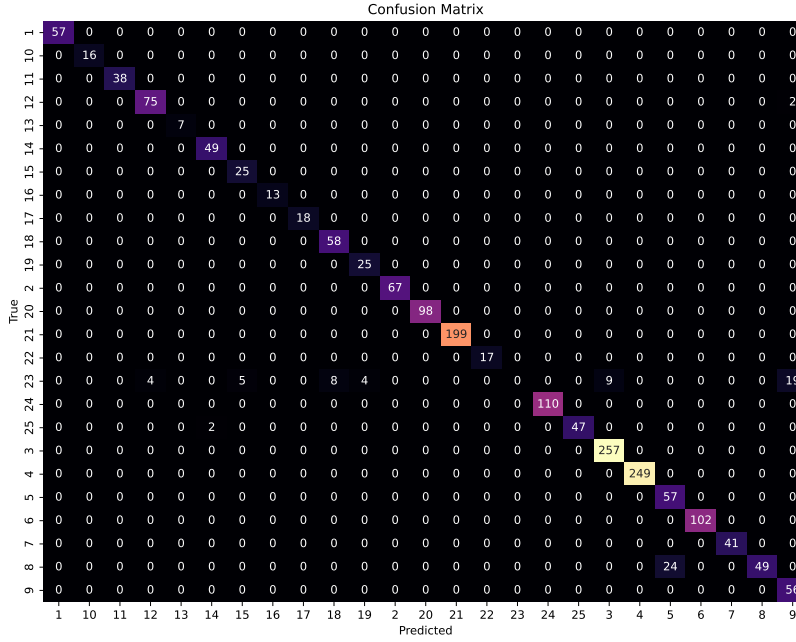


Image 7: Confusion matrix with no misclassifications (25 classes).

Effects of augmentation

During the experimental phase, data augmentation[10] was applied to the training set using techniques such as rotation, mirroring, cropping, and blurring. However, these methods significantly degraded the model’s performance by introducing unnatural distortions that even humans found difficult to recognize. Such augmentations do not reflect real-world conditions, where only slight variations in lighting or angle occur—changes the system can handle reliably. More realistic variations, such as hairstyle changes, facial hair, or makeup, have minimal impact on embeddings and do not impair recognition.

Testing limitations

With 25 classes, there are minor misclassifications on certain similar images, but they are almost negligible, and each class is mostly correctly predicted overall. At 50 classes, the classification drops to 73%. A way to improve the results would definitely be to manually record 50 or more people in 2 videos under different conditions, which would provide higher quality test data and better embeddings. This part will be further explored during the scalability test evaluation. Testing and validation were carried out by comparing predictions on the validation set with images from the training set, rather than real-time live input. Although this evaluation method is necessary, it does not fully reflect the challenges of real-time face recognition from a live feed, where factors such as lighting changes, different facial angles, facial expressions, and resolution variations can further impact model performance, but it also excludes some advantages that live testing provides. Unfortunately, due to limitations in standard experimental protocols, simulating these real-world conditions through the validation set currently represents the closest possible method of testing before implementation in real-time face recognition systems. Using a live feed allows for the analysis of many more consecutive frames in a shorter time, which enables the aggregation of predictions and reduces the impact of certain anomalies that may occur on individual images due to poor lighting, shooting angles, blurriness, etc. Instead of relying on static images, the system can collect more facial samples from the user over time, improving recognition reliability. This approach is particularly useful in dynamic systems where the goal is to achieve stable and precise recognition, such as security systems or contactless authentication.

Furthermore, live input enables adaptive techniques such as face tracking, where the model can continuously correct and improve detection by analyzing micro-movements and different facial expressions. This flexibility often leads to better overall accuracy in real-world environments than would be expected based on validation results obtained from traditional static image testing methods.

FAISS Implementation

Face embeddings were visualized in a two-dimensional space using Principal Component Analysis (PCA)[11], which reduced the dimensionality by transforming multi-dimensional data into a smaller number of principal components, retaining as much variability as possible from the original data and their placement within the Clip embedding space. This approach provided insight into their distribution and mutual relationships. Some overlap between embeddings of different classes was observed, indicating challenges in distinguishing very similar faces. The overlap was especially pronounced for individuals with similar facial features, lighting, or expressions, making simple recognition methods more difficult. The more similar classes there are in the system and the larger the number of their instances, the greater the overlap, making it harder to clearly separate the classes, thus increasing the risk of misclassification, or confusion between classes that are close within the space. Two potential simple solutions would involve reducing the total number of classes or reducing the number of instances per class, but neither option is satisfactory. This situation highlights the need for advanced search methods, such as FAISS, which allows for fast and precise identification despite the overlap of embeddings.

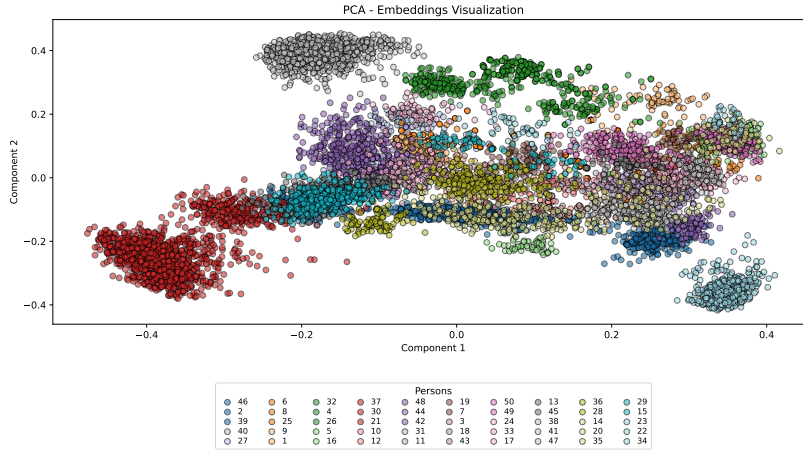


Image 8: Realistic depiction of classes and their overlap

To address the problem of increasing overlap, FAISS (Facebook AI Similarity Search) was introduced, an optimized system for similarity search in large embedding spaces. FAISS uses efficient data structures such as ANN (Approximate Nearest Neighbors) search and quantization to speed up the recognition process and reduce the negative effects of overlap. This allows for more accurate and faster classification of instances, despite the increased data density in the space. The previously mentioned k1 and k2 parameters are specifically related to FAISS and represent how many matches it seeks in each step of embedding classification.

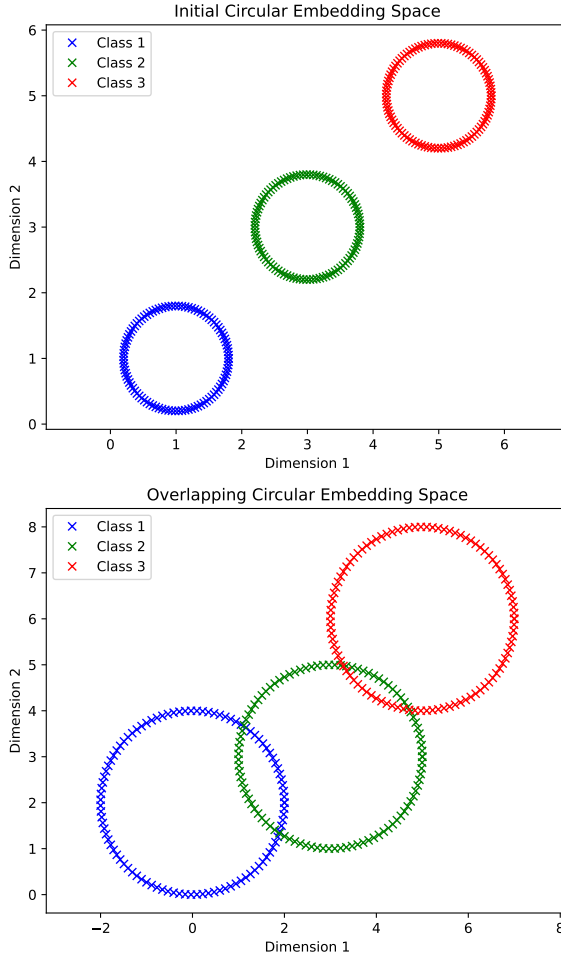


Image 9: Simplified depiction of overlap

Reaching peak performance

After a certain threshold value, the model reaches its maximum classification potential, where further increases in the threshold no longer affect the metrics. This indicates a high confidence in its decisions and the ability to clearly separate positive and negative examples. All positive classifications are above a certain confidence threshold, while negative ones are below it, meaning the model consistently and accurately makes decisions without uncertain predictions in the middle probability range. This effect confirms the robustness of the model and its resistance to threshold changes, achieving optimal accuracy in real-world scenarios. This is especially useful in systems where classification reliability is crucial, and further adjustment can only be used for fine-tuning the model in specific cases. Precision and recall remain the same because FP and FN are always equal within each parameter combination. This means the model consistently confuses certain classes, regardless of the threshold or k-values it considers. This is also evident through the precision-recall analysis.

Precision-recall curve

For each individual parameter combination from the grid search, Precision and Recall are identical because in each combination, the number of False Positive and False Negative predictions is equal. This suggests that the model is making errors on the same set of images (the same classes) regardless of changes in parameters, indicating the presence of problematic samples that are inherently difficult to classify. These results imply that the limitations do not necessarily arise from the model, but from the quality of the data. If these specific images were removed or improved, the model's performance would likely increase significantly.

Furthermore, the fact that Precision and Recall remain equal across all decision thresholds results in a P-R curve[12] resembling a diagonal line, which at first glance may resemble a random

classifier. However, in this case, it is not due to random guessing, but rather a result of an uneven distribution of classification performance. The model achieves high accuracy for certain classes, while for others it fails to differentiate between positive and negative examples. This causes the typical curvature of the P-R curve to disappear, as variations in the threshold do not affect the relationship between TP, FP, TN, and FN, but rather maintain a constant interrelationship that changes proportionally.

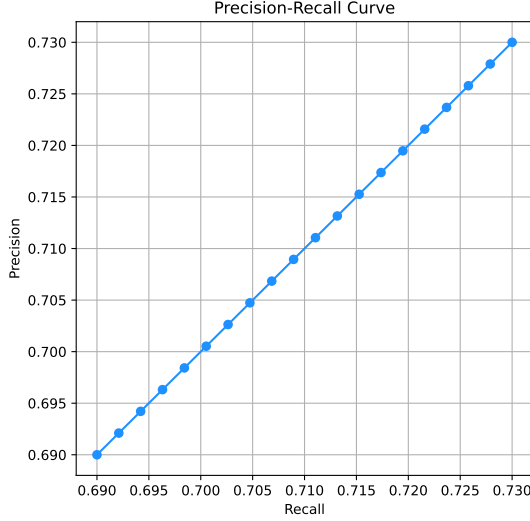


Image 10: Precision-recall curve.

This result emphasizes the need for data analysis and potential improvements in the problematic classes, rather than focusing solely on changes to the model’s hyperparameters. Additionally, if the ”problematic” classes did not exist, Precision and Recall would remain at a constant 100%, resulting in a flat curve.

By examining the graph, it is evident that in each precision-recall pair, both members are equal, which is a clear indicator of the existence of problematic classes. Evaluation of the best point of grid search indeed confirms existence of problematic classes.

Class results and handling of problematic classes

During the grid search process, the best metrics were achieved at a certain configuration. Detailed class evaluation for the best grid search point with 50 classes yielded the following results: (1, 3, 0.82, 0.7305, 0.7305, 0.6770, 2702, 991, 46577, 991) — indicating 991 misclassified images in total. A closer inspection of these results reveals that some classes perform significantly worse than others, which contributes to the overall misclassification rate.

The evaluation shows that some classes achieve excellent results, while others are significantly weaker, thus affecting the overall system performance. Specifically, the table shows the distribution of classes into three performance groups (0-49%, 50-74%, 75-100%).

Class	Precision	Recall	F1-Score	Support
1	0.98	1.00	0.99	57
10	0.31	1.00	0.48	16
11	0.20	1.00	0.33	38
12	0.99	0.97	0.98	77
13	0.88	1.00	0.93	7
14	0.21	0.14	0.17	49
15	0.52	1.00	0.68	25
16	1.00	1.00	1.00	12
17	1.00	1.00	1.00	18
18	0.94	1.00	0.97	58
19	1.00	1.00	1.00	25
2	0.92	0.99	0.95	67
20	1.00	1.00	1.00	98
21	1.00	0.91	0.95	199
22	1.00	1.00	1.00	17
23	0.00	0.00	0.00	49
24	0.40	0.69	0.51	114
25	1.00	0.96	0.98	49
26	0.87	1.00	0.93	13
27	0.95	1.00	0.98	61
28	1.00	0.05	0.10	57
29	0.00	0.00	0.00	57
3	1.00	1.00	1.00	257
30	0.12	0.05	0.07	185
31	0.48	0.69	0.56	58
32	0.99	1.00	0.99	87
33	1.00	1.00	1.00	71
34	1.00	0.09	0.17	173
35	0.51	1.00	0.68	53
36	0.75	1.00	0.85	156
37	0.61	1.00	0.76	35
38	0.62	1.00	0.76	73
39	0.53	0.98	0.69	61
4	1.00	0.98	0.99	249
40	1.00	1.00	1.00	13
41	0.00	0.00	0.00	0
42	1.00	1.00	1.00	45
43	0.00	0.00	0.00	53
44	0.00	0.00	0.00	57
45	0.42	0.93	0.58	83
46	0.62	1.00	0.76	69
47	1.00	0.10	0.18	153
48	0.42	0.95	0.58	77
49	1.00	0.91	0.95	56
5	0.65	0.96	0.78	57
50	1.00	0.04	0.08	142
6	0.92	1.00	0.96	102
7	1.00	1.00	1.00	41
8	0.91	0.67	0.77	73
9	0.98	1.00	0.99	56
Unknown	0.00	0.00	0.00	0

Table 2: Metrics for each class

	Precision	Recall	F1-Score
0-49%	10, 14, 11, 43, 23, 29, 4, 24, 30, 45, 31, 41, 48	14, 30, 44, 23, 34, 47, 28, 41, 29, 50, 43	10, 14, 11, 43, 23, 30, 44, 28, 34, 47, 29, 41, 50
50-74%	15, 5, 35, 37, 38, 39, 46	24, 31, 8	15, 24, 31, 35, 39, 45, 48
75-100%	1, 12, 16, 17, 18, 19, 2, 20, 21, 22, 25, 26, 27, 28, 3, 32, 33, 34, 36, 4, 40, 42, 47, 49, 50, 6, 7, 8, 9, 13	1, 10, 11, 12, 15, 16, 17, 18, 19, 2, 20, 21, 22, 25, 26, 27, 3, 32, 33, 35, 36, 38, 39, 4, 40, 42, 45, 46, 48, 49, 5, 6, 7, 9, 13	1, 12, 16, 17, 18, 19, 2, 20, 21, 22, 25, 26, 27, 3, 32, 33, 34, 36, 4, 38, 42, 46, 49, 5, 6, 7, 8, 9, 13

Table 3: Performance

distribution table

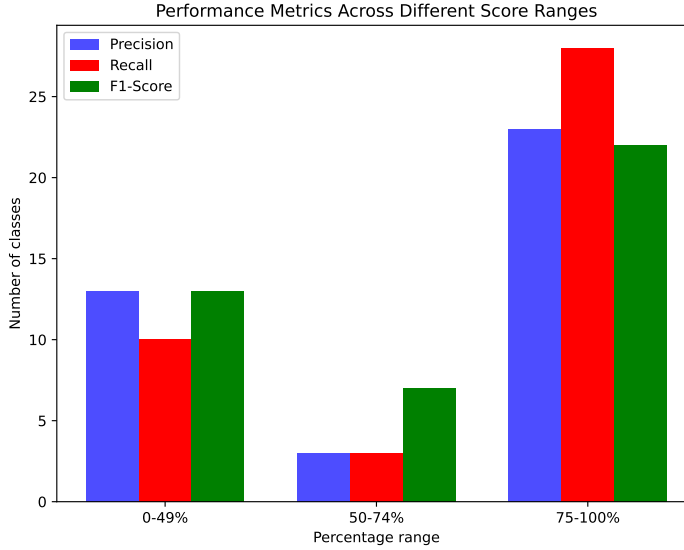


Image 11: Range-based

metrics distribution.

By identifying problematic classes that are quite difficult to classify even by eye due to their low quality, we came up with the idea of evaluating the model without these classes. By removing 17 extremely low-quality classes, the best evaluation combination of parameters for the previous dataset was tested, and the following results were achieved:

- Accuracy: 0.8786 - Recall: 0.8786 - F1 Score: 0.8628 - TP: 2295, FP: 317, TN: 9827, FN: 317

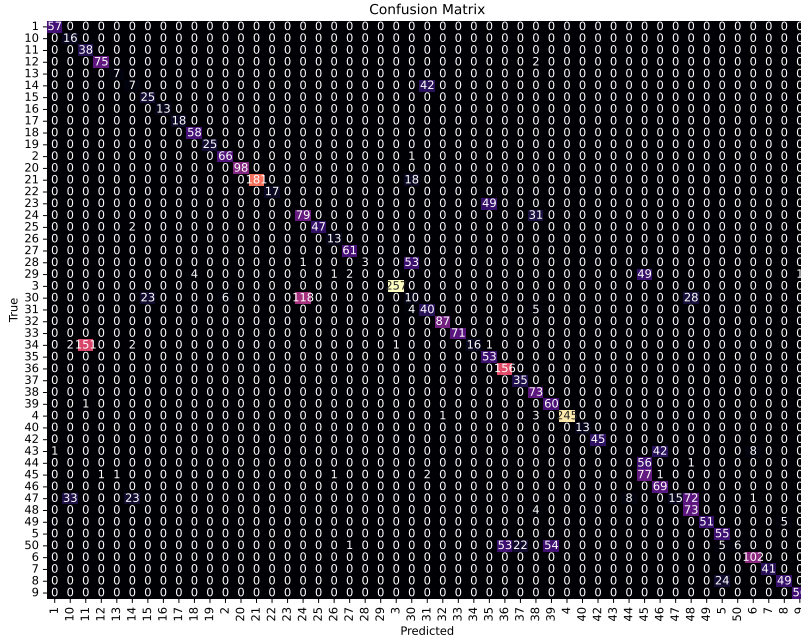


Image 12: Confusion matrix

with problematic classes.

A more detailed analysis of the confusion matrix [13] reveals that there are certain problematic classes that constantly mix with each other. Specifically, these are: 11-34, 39-50, 36-50, 29-49, 48-47, 35-23, 28-30. If we remove some of them, the metrics automatically improve, although not perfectly. It is evident that now the mixes are 36-50, 39-50, 47-6, 34-13, 34-3. Therefore, theoretically, the system could be brought to 100% performance through iterative evaluation and elimination of the classes that mix, but such evaluation would not reflect realistic real-world deployment conditions. However, removing classes with low-quality or inconsistent data has a notably positive impact on the overall system performance.

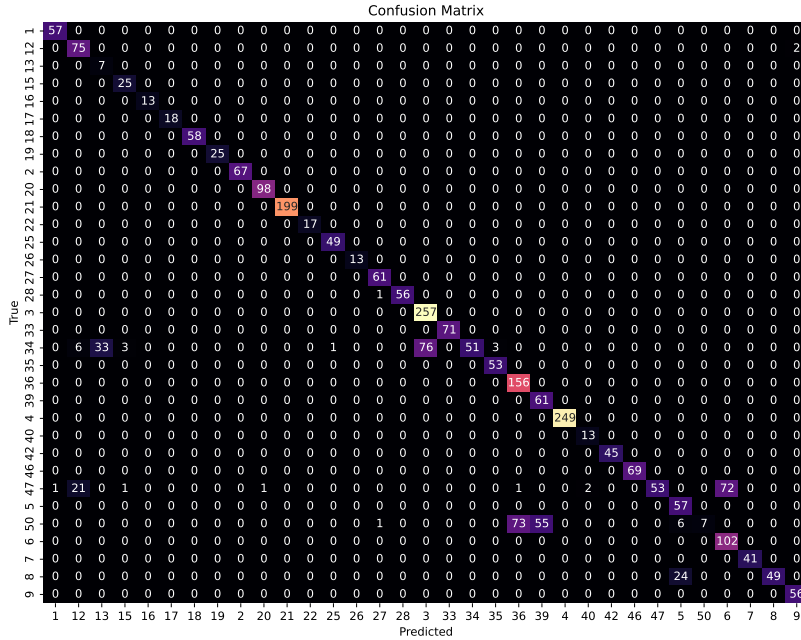






Image 13: Confusion matrix

without problematic classes.

By removing problematic classes from the dataset, there is much less confusion during classification, and the system's performance improves, which is yet another indicator that most of the errors

did not arise from the system's implementation, but from classes whose training and validation images have certain flaws that prevent their correct classification. Of course, metrics such as the ROC curve (compressed to the top left) or the P-R curve (diagonal) will still look similar due to the nature of multi-class classification and the large number of true negatives, i.e., the equality of false positives and false negatives. Next, there are a few examples of train-val pairs:

Class	Images	Description
20		Class 20 achieves perfect metrics with a precision of 1.00, meaning that all samples predicted as class 20 are indeed correct.
17		Class 17 also records excellent results with a perfect precision of 100%, making it flawless in predictions.
46		Class 46 achieves a precision of 0.62, which is decent, but there is room for improvement.
39		Class 39 has a precision of 0.53, which means there is a significant number of false positive predictions.

The "Unknown" class achieves a precision of 0.00, which is actually a good sign because the "Unknown" class is defined for cases where the model doesn't know how to classify a particular image. This indicates that the system will provide an output for each input that, with sufficient data quality, will be accurate.

Image 14: Few class

examples.

ROC curve

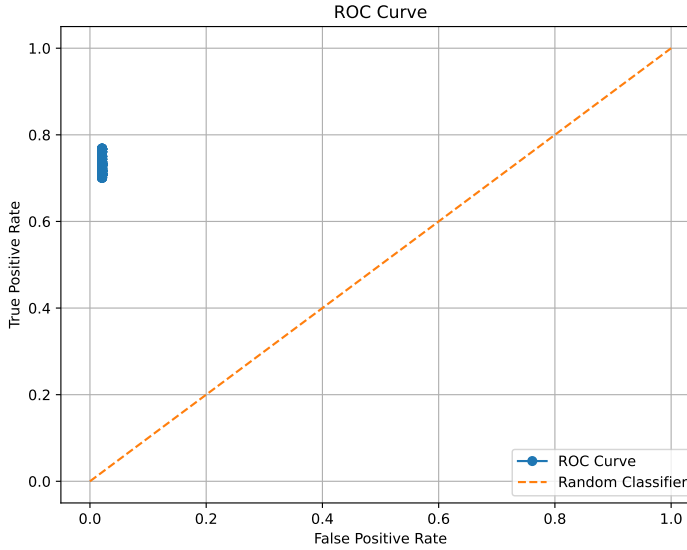


Image 15: Global ROC

curve.

The ROC curve [14] in our case does not provide useful information due to the large imbalance between true negative (TN) and true positive (TP) values. Since the classification involves multiple classes, each positive prediction automatically generates negative predictions for all other classes, resulting in a very low False Positive Rate (FPR) and minimal change in the True Positive Rate (TPR). As a result, the ROC curve is compressed into the top left corner of the graph, where it remains almost stationary. Additionally, the false positive and false negative values are always the same because they are interconnected (incorrectly labeling a sample as belonging to the wrong class automatically leads to incorrectly labeling it as not belonging to the correct class).

Although it does not show progressive changes in TPR, the positioning of the curve in top left corner confirms that the model correctly recognizes positive cases with few false positives. The imbalance arises because for each image, the model must determine 49 times that it does not belong to a class, while it only determines once that it belongs to a specific class.

Time-based scalability test

In testing the system's performance, the effect of increasing the number of classes on the loading time was observed, with a specific focus on the phases of converting images into embeddings and loading them into FAISS. [15] As expected, the loading time did not grow proportionally with the number of classes. Instead of a linear relationship, the system showed disproportionate growth in time (the increase in the number of classes was greater than the increase in execution time), indicating optimized loading processes and scalability. This result suggests that the system can efficiently handle larger numbers of classes without a significant increase in loading time. Expected increase would follow the given formula:

$$\text{Total loading time} = \text{Number of classes} \times \text{Loading time per class} \quad (1)$$

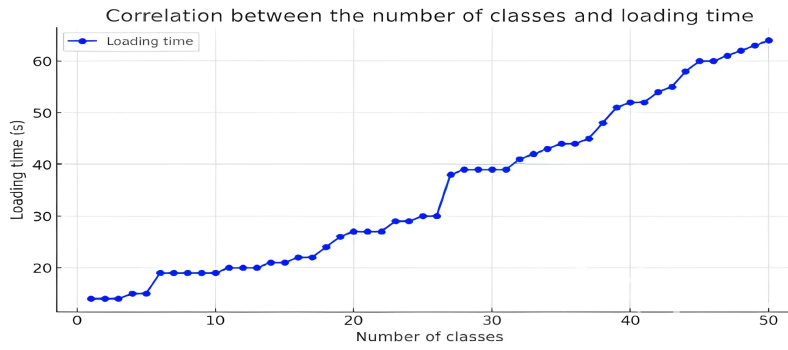


Image 16: Time-based scalability test.

However, the increase was noticeably smaller. For loading the system with 1 class, it took 14 seconds, while for loading the system with 50 classes, it took 64 seconds. Therefore, with an increase in the number of classes by 4900%, the loading time increased by 357.14%.

Metrics-based scalability test

The graph shows how the model's performance depends on the number of classes, with Precision, Recall, and F1-score initially remaining at a high level, and a significant drop occurring only after approximately 30 classes. However, the decline in performance is not necessarily only related to the increase in the number of classes, but also to the quality and diversity of the images in the dataset. It is important to note that Precision and Recall consistently align, indicating a balance between the model's accuracy and completeness of recognition. Although all three metrics decrease with the increase in the number of classes, data quality plays a crucial role in maintaining good results. With a well-prepared dataset, the model can maintain high performance even with a larger number of classes. Moreover, by adding high-quality classes, it is possible to improve the metrics, suggesting that the structure, quality, and informativeness of the data has a greater impact on system performance than the number of classes alone.

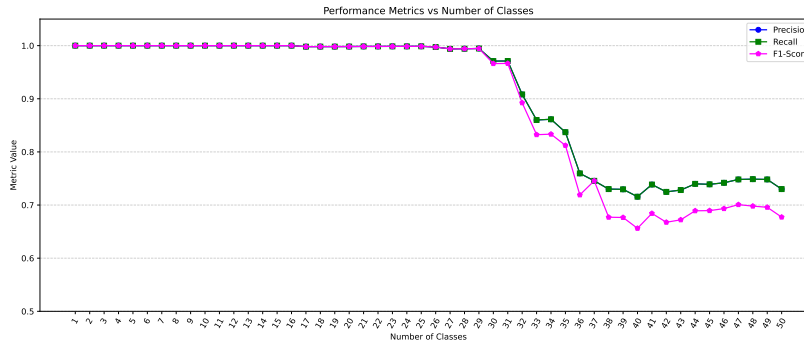


Image 17:

Performance-based scalability test.

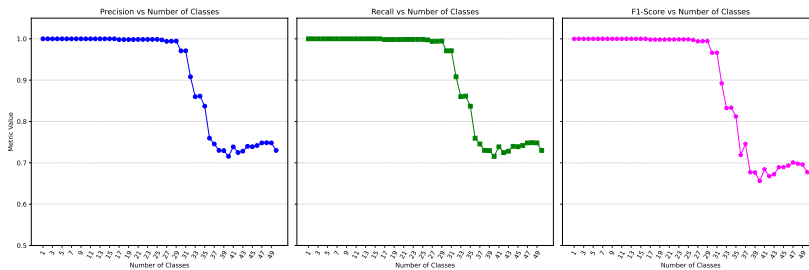


Image 18: Separated plots.

Model	Architecture	Key metric	Accuracy	Pros	Cons
FaceNet (Google)	CNN + triplet loss	Cosine similarity	99.63%	Small embeddings (128 dim), robust	High training resource usage
DeepFace (Facebook)	CNN (9 layers)	Euclidean + cosine	97.35%	First human-level model	Hard to deploy and optimise
VGG-Face	CNN (37 layers)	Euclidean	98.95%	Deep architecture improves accuracy	Many parameters, slow inference
ArcFace	CNN + Arc loss	Cosine similarity	99.40%	Better class separation	Needs fine-tuned models
YOLO-Face	YOLO CNN	IoU	99.8%	Fast all-in-one recognition	Weaker on complex faces
Our system (CLIP + FAISS)	CLIP + FAISS search	L2/Cosine	~87–100% depending on dataset	Fast, scalable, no training required	Embedding quality affects performance

Table 4: Comparison of face recognition models

Comparison with similar solutions

Compared to similar solutions,[16] our system has certain advantages. Firstly, it does not need to be trained in the traditional sense; new classes/new faces can be added very quickly and easily. Furthermore, the system is extremely easy to implement in applications that involve various forms of face detection without requiring excessive modification, as the operating principle remains unchanged. Additionally, due to CLIP’s strong capabilities in extracting embeddings, with a quality dataset, it enables successful recognition of an exceptionally large number of different faces, especially if the conditions for capturing reference images to be added to the system are predefined. It is particularly interesting that successful recognition does not require the live detection input to capture the entire face. The system is not limited by head coverings, masks, etc. If a face is still sufficiently recognizable even with such coverings, the system will classify it correctly. The key is that the system successfully identifies specific parts of the face that contribute most to distinguishing an individual from others.



Image 19: Example for live detection for covered face.

Real-life applications

The described system can be implemented for real-time face detection and recognition using a computer camera by passing camera frames as input, creating a bounding box around each detected person, extracting embeddings, and passing them into the described system. The predicted most likely class (with the most votes) is returned as the function’s output and added as a label on the box. Additionally, the system can handle multiple known faces simultaneously in the same frame, which adds an extra dimension to the system and enables more advanced recognition in complex situations. It is also important to note that faces can be classified in challenging conditions such as low-light environments. This implementation provides numerous possibilities, such as creating systems for student attendance tracking, payroll calculation based on hours worked, security systems, various interactive interfaces, and more. The system’s latency is under 50 ms, and it can process around 200 frames per second, which is an indicator of fairly good performance.

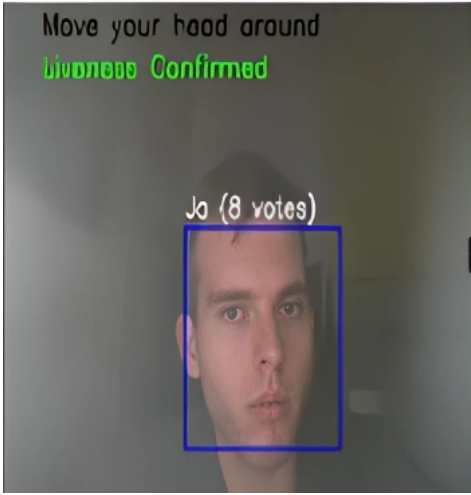


Image 20: Example for live detection in dark surroundings.



Image 21: Example for live detection of multiple faces.

Limitations and conclusions

Face recognition analysis has shown that image quality and background significantly impact the model's accuracy. When backgrounds are too varied, embeddings become less precise, making it harder to correctly recognize individuals. Additional challenges arise from text, objects in the frame, and large lighting changes, as these introduce variations that are not related to the identity of the person but to the environment.

This problem is particularly evident when the system works with a large number of different faces — while the model achieves high accuracy under controlled conditions, its performance may decline if the data varies in lighting, capture angle, or image quality. Factors such as blurring, low resolution, and contrast can further reduce the reliability of recognition. To improve results, it would be beneficial to standardize the conditions for capturing reference images or apply methods that ensure greater consistency in input data. For example, in a system for automated student recognition at a university, it would be advisable for all reference images to be captured under the same conditions and with a uniform background, which would neutralize recognition errors.

In conclusion, the results show that the system does not have issues with face recognition, but its accuracy is most affected by external factors that can degrade the quality of the embeddings.

Unlike traditional methods[17] that rely on deep convolutional networks specialized for faces, our system uses multimodal embeddings, offering greater flexibility and greater application in various recognition scenarios. Furthermore, the integration of FAISS indexing allows for extremely fast searches, significantly improving the efficiency and responsiveness of the model in real-time. Thus, this work contributes to the field of computer vision, presenting a new methodology that combines the power of transformer embeddings and optimized vector search, opening opportunities for future research and improvements in fast and scalable face recognition.

References

- [1] Songze Zhu. Enhancing facial recognition: A comprehensive review of deep learning approaches and future perspectives. *Applied and Computational Engineering*, 110:137–145, 11 2024. Accessed: 2025-04-19.
- [2] Divyarajsinh N. Parmar and Brijesh B. Mehta. Face recognition methods & applications, 2014. Accessed: 2025-04-19.
- [3] Facebook AI Research. Faiss: A library for efficient similarity search and clustering of dense vectors. <https://faiss.ai/>, 2025. Accessed: 2025-04-16.
- [4] Nhan T. Luu. Clip unreasonable potential in single-shot face recognition. *arXiv preprint arXiv:2411.12319*, 2024. Accessed: 2025-04-19.
- [5] Jason Brownlee. Distance measures for machine learning. <https://machinelearningmastery.com/distance-measures-for-machine-learning/>, 2019. Accessed: 2025-04-24.
- [6] Lior Wolf, Tal Hassner, and Itay Maoz. Youtube faces database. <https://www.cs.tau.ac.il/~wolf/ytfaces/>, 2011. Accessed: 2025-04-24.
- [7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [8] NumPy Developers. numpy.linalg.norm — numpy v1.26 manual, 2024. Accessed: 2025-04-24.
- [9] David M. W. Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011. Archived from the original on 2019-11-14, Accessed: 2025-04-24.
- [10] Suorong Yang, Weikang Xiao, Mengchen Zhang, Suhan Guo, Jian Zhao, and Furao Shen. Image data augmentation for deep learning: A survey. *arXiv preprint arXiv:2204.08610*, 2022.
- [11] Jake Lever, Martin Krzywinski, and Naomi Altman. Principal component analysis. *Nature Methods*, 14(7):641–642, 2017. Accessed: 2025-04-19.
- [12] Rohit Kundu. Precision vs. recall: Differences, use cases & evaluation, 2022. Accessed: 2025-04-24.
- [13] GeeksforGeeks. Understanding the confusion matrix in machine learning. <https://www.geeksforgeeks.org/confusion-matrix-machine-learning/>. Accessed: 2025-04-22.
- [14] Zhe Hui Hoo, Jane Candlish, and Dawn Teare. What is an roc curve? *Emergency Medicine Journal*, 34(6):357–359, 2017. Accessed: 2025-04-24.
- [15] Erik Scepanski and Sonja Zillner. Ai systems and their scalability – a systematic literature review. In *Proceedings of the 35th Australasian Conference on Information Systems (ACIS 2024)*, 2024.
- [16] Thinking Neuron. Face recognition using deep learning (cnn) in python. <https://thinkingneuron.com/face-recognition-using-deep-learning-cnn-in-python/>, 2020. Accessed: 2025-04-25.
- [17] Ena Barcic, Petra Grd, and Igor Tomicic. Convolutional neural networks for face recognition: A systematic literature review, 07 2023.