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

April 2025

napomena: ne znan štimat slike, s tin ću se još malo poigrat

Abstract

Face recognition is increasingly being adopted across industries like 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 like 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, implementing 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 high accuracy exceeding 90% and 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. 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. 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, relying 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) and CLIP (Contrastive Language-Image Pretraining) 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 for handling 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. Additionally, 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 individually been applied across various domains, there has been no recorded implementation 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, where each class represents 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, which is later used for recognition.

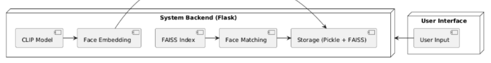


Image 1: Schema overview.

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 L2 distance, also known as Euclidean distance.

The L2 distance between two vectors A = (a1, a2, ..., an) and B = (b1, b2, ..., bn) 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—the smaller the distance, the more similar the 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 labeled as "Unknown".

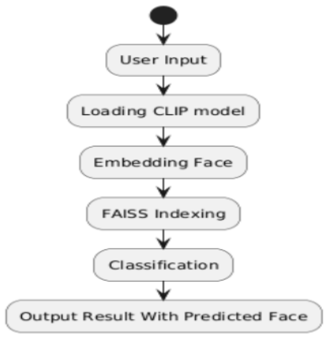


Image 3: System flowchart.

CLIP embedding example (face image for input and corresponding face embedding for output):

Experiments, tests and results

First, the system was tested on a dataset of about 500 classes, where each person had 1 video, using some

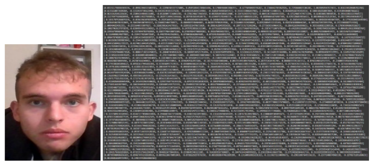


Image 4: Embeddings example.

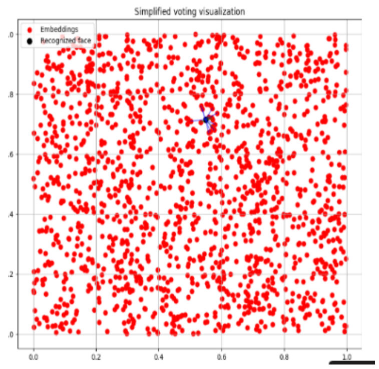


Image 5: Simplified voting visualization.

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 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 6: An example of train image.

Image 7: An example of val image.

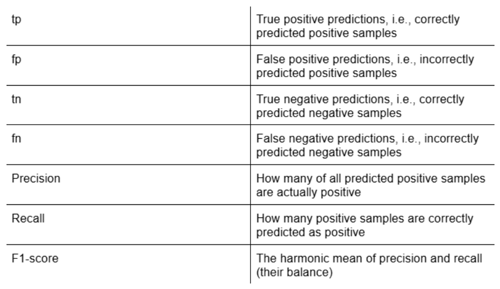
Evaluation process

After defining the dataset, an extensive procedure was carried out to test various combinations of parameters k1, k2, and threshold. A grid search was used for efficient parameter search. The following parameter ranges were defined: k1: 1-9 k2: 2-10 threshold: 0.5 - 0.1 with a step of 0.01 Additionally, a condition was defined to skip combinations where k1 and k2 are equal and where k2 is smaller than k1, in order to avoid meaningless steps in the search. The testing and evaluation process was carried out by forwarding each image from all the validation folders to CLIP to create its embeddings. Then, the embeddings were normalized for easier comparison using linalg.norm.

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 the image belongs to 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: k1, k2, threshold, precision, recall, f1, tp, fp, tn, fn.



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)

Image 8: Metrics overview.

In the experimental phase, data augmentation for the training set was tested using techniques such as mirroring, rotation, cropping, blurring, etc. This approach significantly worsened the model's performance and deviated the evaluation method from the intended real-life use case. Namely, recognizing over-augmented images is quite difficult even with the naked eye, and it also has too much of an impact on the embeddings, making the distances too large for a valid match. Classic augmentation techniques (rotation, cropping, mirroring, blurring, etc.) were tested, but they were completely useless in real-life face recognition scenarios. In real-world situations, a person will not have a distorted or artificially modified face, as such augmentations are impossible in a real-life application of the system. The only real variations are changes in lighting and slight changes in the angle of the shot, which the system can still recognize. However, augmentations beyond that, such as rotation and contrast changes, lose their meaning. In the real world, more realistic "augmentations" include changes in hairstyle, growing a beard, makeup, etc., but these have little impact on the embeddings, as they don't significantly alter the key features of the face.

OVAJ DAN JE PRELIP DA CILI DAN TIPKAN PO LATEX-u...nastavit ću večeras :)