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

April 2025

napomena: ne znan štimat slike i njihove captione, 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.

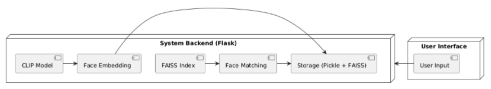


Image 1: Schema overview.

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".

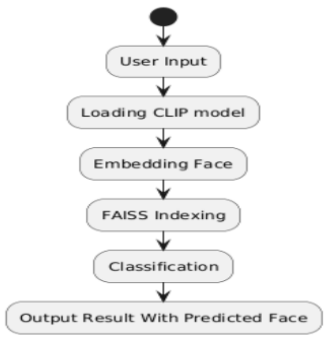


Image 3: System flowchart.

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 frames for training and the rest for validation (80-20 split). In this case, the results showed 100% accuracy

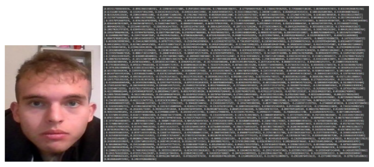


Image 4: Embeddings example.

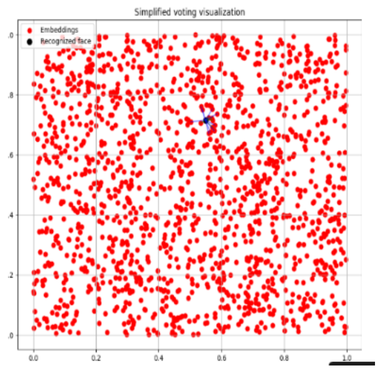


Image 5: Simplified voting visualization.

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.

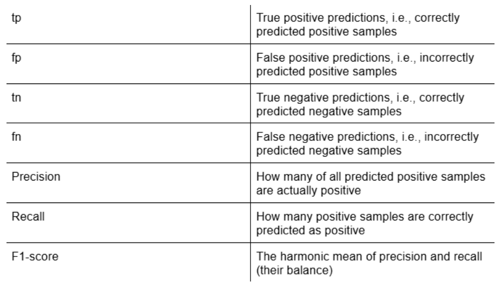
Evaluation process

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.

Example of train image and its valid matching validation image:

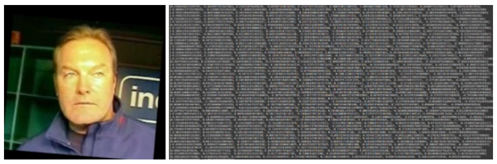


Image 9: Train example.

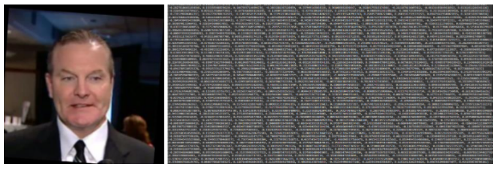


Image 10: Val example.

Image 11: Example of image with applied augmentation.

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.

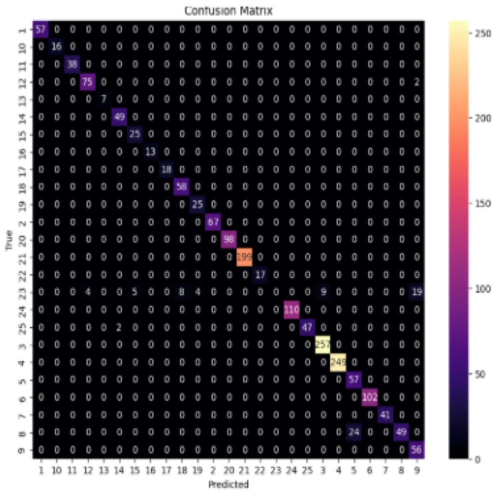


Image 12: Confusion matrix with no misclassification (25 classes).

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), 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, which makes 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.

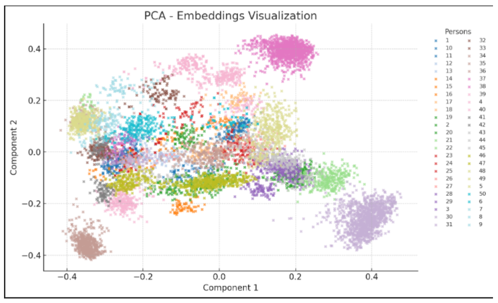


Image 13: Realistic depiction of classes and their overlap

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.

Reaching peak performance

After a certain threshold value, the model reaches its maximum classification potential, where

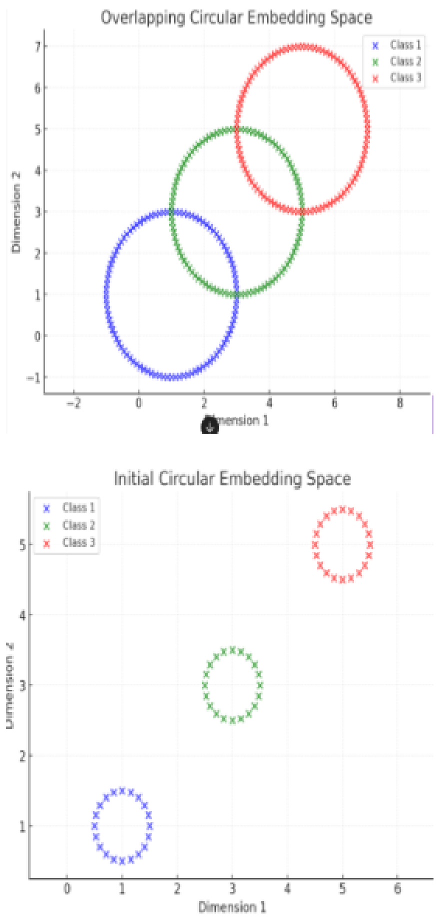


Image 14: Simplified depiction of overlap

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 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.

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 existance of problematic classes.

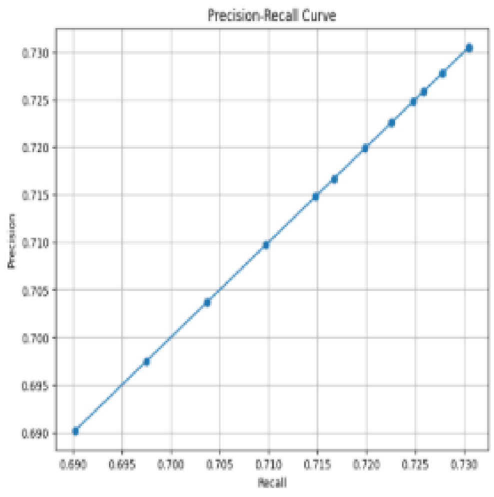


Image 15: Precision-recall curve.

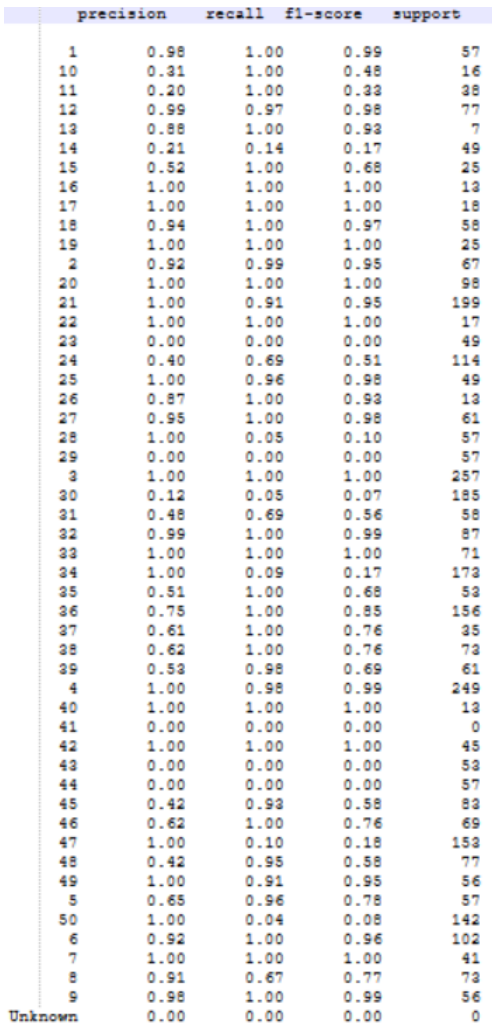
During the grid search process, the best metrics were achieved at a certain configuration. Detailed class evaluation for the best grid search point with 50 classes yielded the following results: (1, 3, 0.82, 0.7305, 0.7305, 0.6770, 2702, 991, 46577, 991) — indicating 991 misclassified images in total. 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%).

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

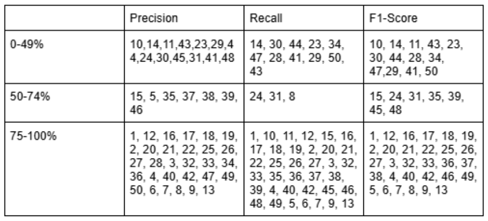
A more detailed analysis of the confusion matrix reveals that there are certain problematic classes that constantly mix with each other. Specifically, these are: 11-34, 39-50, 36-50, 29-49, 48-47, 35-23, 28-30. If we remove some of them, the metrics automatically improve, although not perfectly. It is 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 this would not be a realistic evaluation. However, removing classes that are simply bad has a significant positive impact on the system.



```
                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        13
      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
```

Image
16: Overview of metrics for each class.



	Precision	Recall	F1-Score
0-49%	10,14,11,43,23,29,44,24,30,45,31,41,48	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, 37, 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, 36, 37, 38, 4, 40, 42, 46, 49, 5, 6, 7, 8, 9, 13

Image
17: Performance distribution table.

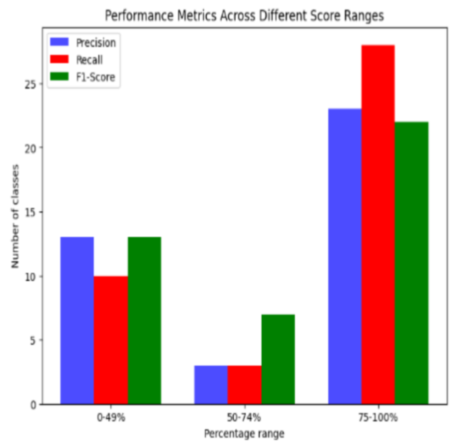


Image
18: Range-based metrics distribution.

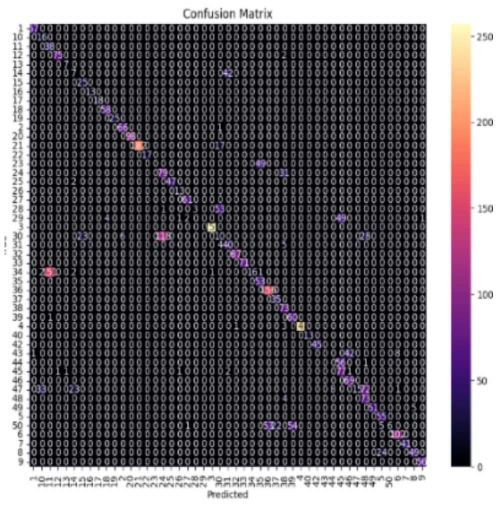


Image
19: Confusion matrix with problematic classes.