UNIVERSITY OF INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

FINAL REPORT

THE ADVANCE COMPUTER VISION

Movie Character Retrieval

Ho Chi Minh City, 01/2025

Acknowledgement

Our group would like to express our heartfelt gratitude to **Dr. Mai Tien Dung**, a lecturer at the Department of Computer Science at the University of Information Technology, Vietnam National University - Ho Chi Minh City. As the instructor for the **CS331.P11.KHTN - The advance computer vision**, his dedicated teaching and insightful feedback have been invaluable to the success of our project.

Throughout the process of completing this project, we strived to deliver the best possible results. However, despite our efforts, some unintentional errors may have occurred. We respectfully request candid feedback and constructive suggestions from the professor to help us further improve our work.

For any questions or feedback, feel free to contact us by email at the following addresses:

- 22521333@gm.uit.edu.vn (Nguyen Duy Thang)
- 22521392@gm.uit.edu.vn (Nguyen Tran Duy Thien)
- 22520002@gm.uit.edu.vn (Tran Kim Ngoc Ngan)

We greatly appreciate your time and guidance!

No	Fullname	Student ID	Contribution	Percent
1	Nguyen Duy Thang	22521333	Extract feature without face, remove duplicated and save index.	40%
2	Nguyen Tran Duy Thien	22521392	Extract feature with face and build demo.	30%
3	Tran Kim Ngoc Ngan	22520002	Build system and merge results.	30%

Table 1: Team members

Contents

1 Reasons for the problem

The problem of retrieving characters in movies has many practical applications and is also an important component in face-based video retrieval systems (Qiao et al. 2020; Arandjelovic and A. Zisserman 2005; Arandjelović and Andrew Zisserman 2006).

Most problems of summarizing important scenes and containing keyfact events of a character in a movie today use faces to identify videos containing characters based on the similarity between the character's face image and the face images in the videos (Standards and (NIST) 2022).

2 Problem statement

Problem description: Given a movie and character sample images in the movie, identify the shots in which the characters appear.

Input:

- Collection of character images - queries.
- A movie - database.

Output: Set of shots that contain the character in the input movie.

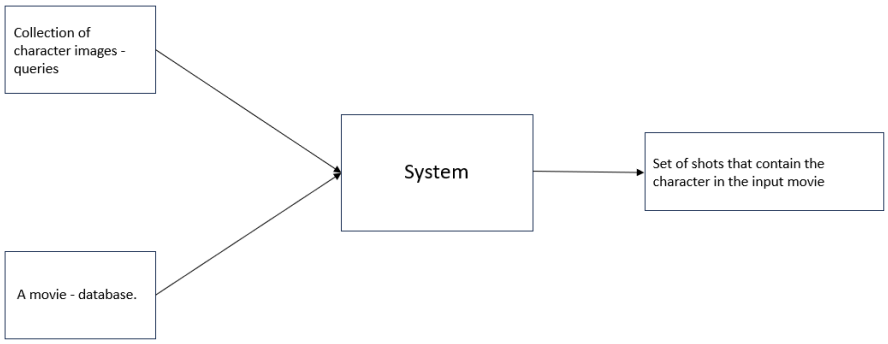

Figure 1: Simple statement

3 Solutions

3.1 Overview

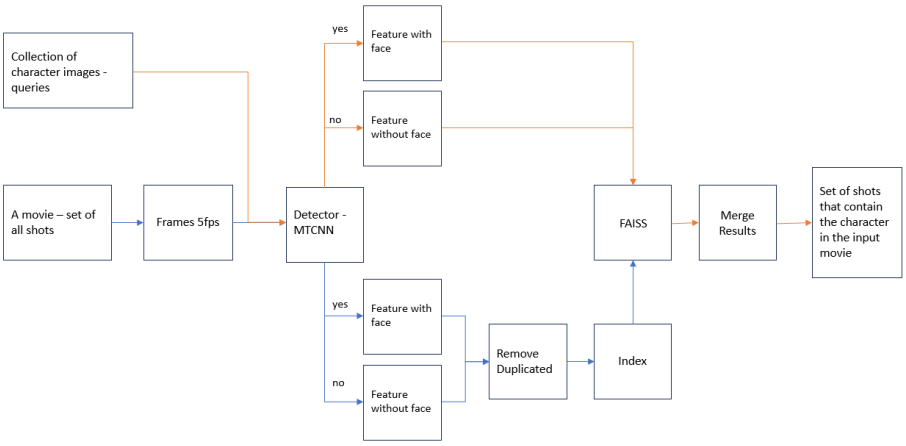

Figure 2: System overviewing

3.2 Data processing

- Each movie consists of multiple scenes, and each scene is divided into shots. For each shot, extract 5 frames per second (the extracted frames may contain human faces or not).
- Use MTCNN to detect faces in each frame. Then, extract features from frames that contain faces as well as those that do not.
- MTCNN is a widely used CNN-based algorithm for face detection and facial landmark localization in images.

Figure 3: MTCNN Example

3.3 Features extracting

3.3.1 Frames with Faces

Each character in the movie has a distinct face, so we can use face recognition methods to identify the characters in the movie.

Use DeepFace (Serengil and Ozpinar 2024) library to extract faces features.

- Author: Facebook AI Research
- Link: https://github.com/serengil/deepface
- DeepFace detects faces in the input image using a pre-trained face detector and extracts high-dimensional feature vectors for each face using pre-trained deep learning models.
- Using MTCNN as the detector, Facenet and ArcFace as the deep learning models

FaceNet (Schroff, Kalenichenko, and Philbin 2015) FaceNet is a facial recognition system that uses deep neural networks to map faces into a vector space such that similar faces have embeddings (vectors) that are close together, while dissimilar faces have embeddings that are far apart.

- Author: Florian Schroff, Dmitry Kalenichenko, James Philbin
- Link: FaceNet: A Unified Embedding for Face Recognition and Clustering

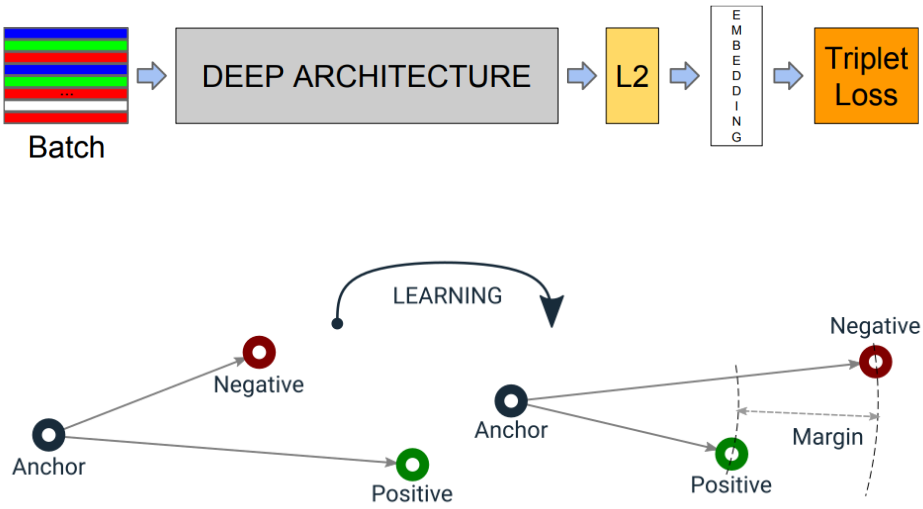

Figure 4: Facenet and triplet loss

Triplet loss

- Anchor (A): The reference data point
- Negative (N): a sample that has a label different from anchor
- Positive (P): a sample that has the same label with anchor

The distance from the anchor to the positive is minimized, and the distance from the anchor to the negative input is maximized. Loss function:

$$\mathcal{L}(A, P, N) = \max\left(0, \|f(A) - f(P)\|_2^2 - \|f(A) - f(N)\|_2^2 + \alpha\right)$$

- If the algorithm is trained with too many "easy" triplets, the model may end up with a poor solution that doesn't work well in real-world situations.
- To solve this problem, hard triplets are used.
  - Hard Positive (P): The positive image should be the farthest from the anchor within the same class. This makes the positive harder to distinguish and forces the model to learn better embeddings.
  - Hard Negative (N): The negative image should be the closest to the anchor, but from a different class, helping the model learn discriminative features for different identities.

ArcFace (Deng et al. 2019) The core idea behind ArcFace is the introduction of an Additive Angular Margin Loss to improve the quality and discriminative power of face embeddings in deep face recognition.

- Author: Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou
- Link: ArcFace: Additive Angular Margin Loss for Deep Face Recognition
- The ArcFace loss builds upon the traditional softmax loss but modifies it by introducing an additive angular margin.
  - Softmax loss:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^{\top}x_i}}{\sum_{j=1}^{C}e^{W_j^{\top}x_i}}$$

  - ArcFace loss:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\cdot\cos(\theta_{y_i}+m)}}{e^{s\cdot\cos(\theta_{y_i}+m)}+\sum_{j\neq y_i}e^{s\cdot\cos\theta_j}}$$

    Where:
    - N is the number of samples in a batch.
    - W_j is the weight vector for class j.
    - x_i is the input feature vector for the i-th sample.
    - y_i is the true class label for the i-th sample.
    - C is the total number of classes.
    - s is a scaling factor.
    - m is the additive angular margin.
    - $\cos(\theta_{y_i})$ is the cosine similarity between the feature vector and the weight vector of the true class.

    The only difference between the Softmax loss and the ArcFace loss is that ArcFace loss uses $cos(\theta)$ (the cosine of the angle between the weight vector and the feature vector) instead of the dot product $W^T x$

    The dot product $W^T x$ can be expressed as:

$$W^T x = ||W||.||x||.\cos(\theta)$$

    To simplify the formula and focus on the angular relationship, normalize W and x to have unit norms:

$$W^T x = \cos(\theta) \text{ (W and x are unit vectors)}$$

    The feature vectors are distributed on a hypersphere with a radius of s. A margin is added to the angle to make the intra-class angles smaller and the inter-class angles larger.

3.3.2 Frame without Faces

Since a query may contain a character but the face cannot be detected, it is important to extract other features from the query as well as from the database.

In a movie, a frame may not only lose facial information but also contain various factors such as background, lighting, costumes, and props. Therefore, a model is needed that can capture all relevant features from the image data and understand the overall contextual meaning of the image, even linking to a broader context.

BEiT (Bao et al. 2021) "Beit: Bert pre-training of image transformers" is a model based on the transformer architecture, designed to learn features from images in a manner similar to how language models like BERT learn from text data. Since this architecture has proven effective in handling global context in text data, it is likely to be effective when applied to understand the overall context of images.

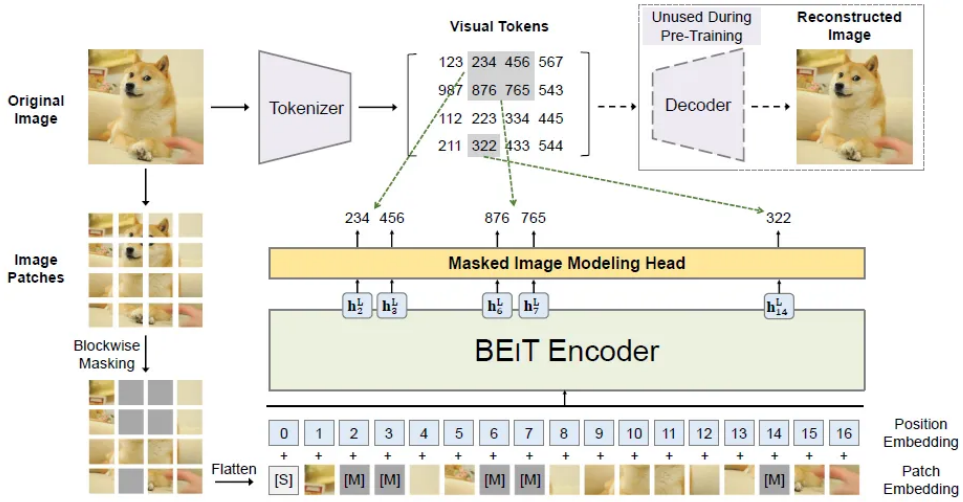

Figure 5: BEiT

- The image is divided into two 'views':
  - First: Visual Tokens: The original image is passed through a Tokenizer to create discrete tokens (the size will be similar to the image after being divided into patches).
  - Second: Image Patches: The original image is divided into patches of similar size.
- The image patches are randomly masked (through Blockwise Masking).
- Then, each patch is flattened and passed through a linear layer to create patch embeddings (similar to word embeddings) – vectors with a fixed length.
- The [S] patch is added at the beginning with the purpose of learning the general representation of the image.
- Position embeddings are added to ensure position information for the patch embeddings.
- The patch embeddings are passed through a backbone Transformer (which becomes the BEiT Encoder after pre-training) to form the vectors
- These vectors are passed through the Masked Image Modeling Head and learn to represent the corresponding visual tokens (the visual tokens are used as targets in the Masked Image Modeling task).
- The model is trained by optimizing the prediction error between the predicted visual tokens and the actual visual tokens.
- Fine-tuning and other tasks: the MIMH layer is replaced with the layer needed. In this task, our team will use the output of the BEiT Encoder as features for a frame. However, BEiT has a special patch, the [S] patch, which will be used in two ways: using the [S] patch as a feature and averaging the patches to form a feature.

CLIP (Radford et al. 2021) (Contrastive Language-Image Pre-training) is a powerful model developed by OpenAI that uses both language and images to understand and represent information. A key component of CLIP is the Image Encoder, which is responsible for converting images into vectors (embeddings) to be compared with the language vectors generated by the Text Encoder.

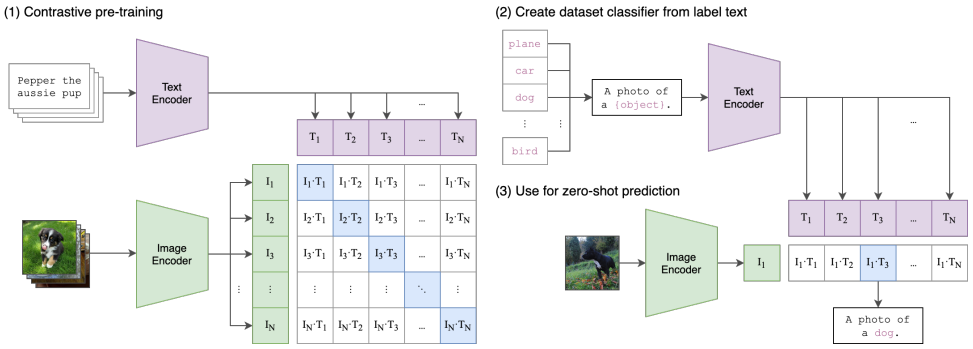

Figure 6: CLIP

- For images, the image is passed through an Image Encoder (ResNet or Vision Transformer) to extract the feature vector I.
- For text, the text is passed through a Text Encoder (CBOW or Text Transformer) to extract the feature vector T.
- Both sets of vectors are mapped into the same space through their respective matrices W, and then normalized.
- The dot product between the vectors I and T is calculated (this is the cosine similarity because the vectors have been normalized).
- The loss is computed and trained such that the values on the diagonal are large, and the off-diagonal values are small.
- In this part, our team will use the pre-trained Image Encoder to extract features. However, according to research, there are two main types of backbones: ViT and ResNet, so our team will proceed with both methods: ViT-B/32 and RN101.

To avoid situations where an image contains a face but it is not detected, our team will also apply both types of features to frames that contain faces as a precautionary measure. So the results may differ from the presentation in class.

3.4 Building indexes for system

Since the frames cut from the same shot are often very similar to each other, the team performs a reduction of these frames:

- Method 1: Perform a check, and if two consecutive frames have a high similarity, discard the later frame.
- Method 2: For the detected faces and feature extraction at each shot, perform clustering using DBSCAN to group the faces of the same person into a cluster.

To perform quickly and accurately, our team will build the system based on FAISS (Douze et al. 2024), a library that helps with efficient querying.
Select the most distinctive feature vectors from each shot to store as an index, which will speed up subsequent runs.

3.5 The system

The system receives the following inputs:

- Movie name.
- Name of the character to query.
- Sample images of the character.
- Type of features the user wants to use.
- Top k results to be returned.

For each query:

- Extract the features of the character's image: If a face is detected, use facial features (ArcFace, FaceNet); if not, use other features (BEiT, CLIP).
- Compare each vector stored in the corresponding index using cosine similarity to find the shots with feature vectors that are highly similar to the query and return the top k results.

Combine the top-k results to create a single final result:

- Sort the top-k by frequency: Shots that appear more frequently will be ranked higher than those that appear less frequently.
- Sort the top-k by similarity: The results will also include the similarity score of each shot with respect to the queries. The shots will be sorted based on this similarity score (if a shot has multiple similarity scores from different queries, the highest score will be used).
- Sort by ranx (Bassani and Romelli 2022): A library is used to combine results using different methods. The chosen method is RRF (Cormack, Clarke, and Büttcher 2009) (Reciprocal Rank Fusion).

4 Experimental results

4.1 Metrics, evaluation

Precision@k: The proportion of query results returned that are correct relative to the total number of results returned.

$$Precision@k = \frac{relevance\_retrieved\_results}{retrieved\_results} = \frac{true\_positive@k}{true\_positive@k + false\_positive@k}$$

Recall@k: The proportion of correct results found relative to the total number of actual correct results.

$$Recall@k = \frac{relevance\_retrieved\_results}{relevance\_results} = \frac{true\_positive@k}{true\_positive@k + false\_negative@k}$$

Precision-Recall Curve for each query: For each different query threshold level, the system will return different shots, leading to changes in Precision and Recall. The Precision-Recall Curve shows these changes as threshold changes.

Mean Average Precision (MAP) for all queries: AP is the average of Precision at different Recall levels, providing a measure of the overall performance of a query. MAP is the average of AP on many different queries. MAP helps to evaluate the performance of the system over the entire query set. The higher the MAP, the better the system performs.

4.2 Dataset

5 movies used in ACM Multimedia DVU Grand Challenges and TRECVID DVU 2023: Calloused Hand, Like Me, Losing Ground, Memphis, Liberty Kid.

The movies have been cut into scenes and shots and have ground-truth available.

4.3 Experimental process and Results

Because there are many steps to combine and each step has many ways to do it, our team will do it on one character first to be able to choose the best way (according to that character) at each step, then change the ways to consider the change in results.

Query: Byrd - Calloused Hands

- k = 500, Feature with face: Facenet, Feature without face: CLIP (rn), Remove type: DBSCAN, Merge Result: No.

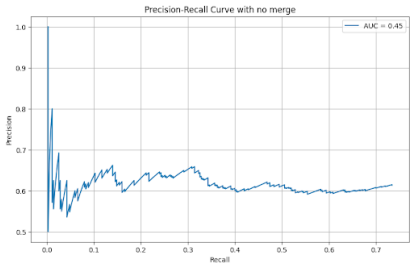

Figure 7: Precision-Recall Curve on first try.

- k = 500, Feature with face: Facenet, Feature without face: CLIP (rn), Remove type: DBSCAN, **Merge Result**: Yes.

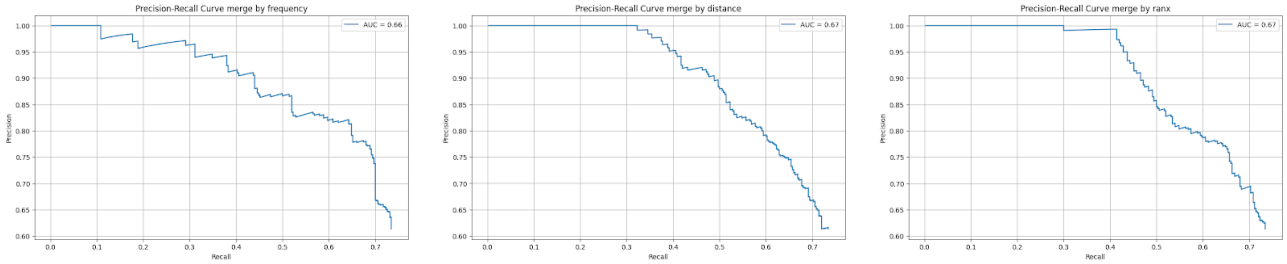

Figure 8: Precision-Recall Curve on second try.

- k = 500, Feature with face: Facenet, Feature without face: CLIP (rn), **Remove type**: Compare, Merge Result: Yes.

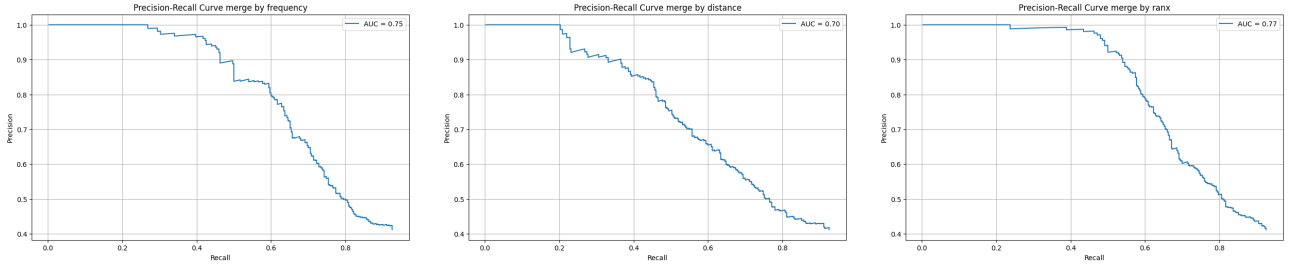

Figure 9: Precision-Recall Curve on third try.

- **k** = 5000, Feature with face: Facenet, Feature without face: CLIP (rn), Remove type: Compare, Merge Result: Yes.

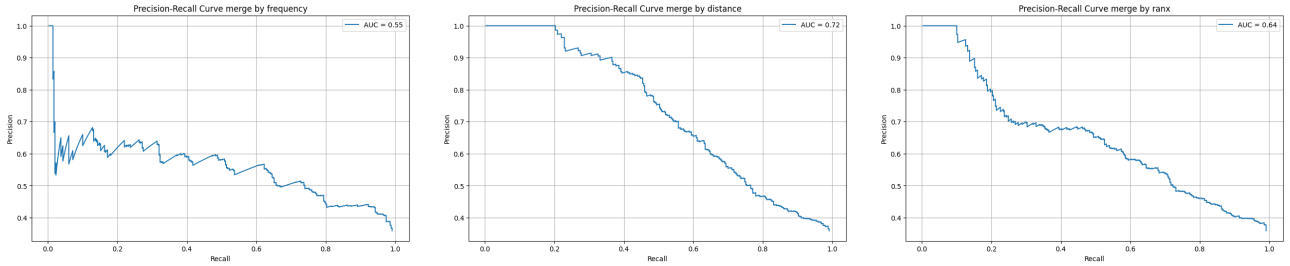

Figure 10: Precision-Recall Curve on fourth try.

- k = 5000, **Feature with face**: ArcFace, Feature without face: CLIP (rn), Remove type: Compare, Merge Result: Yes.

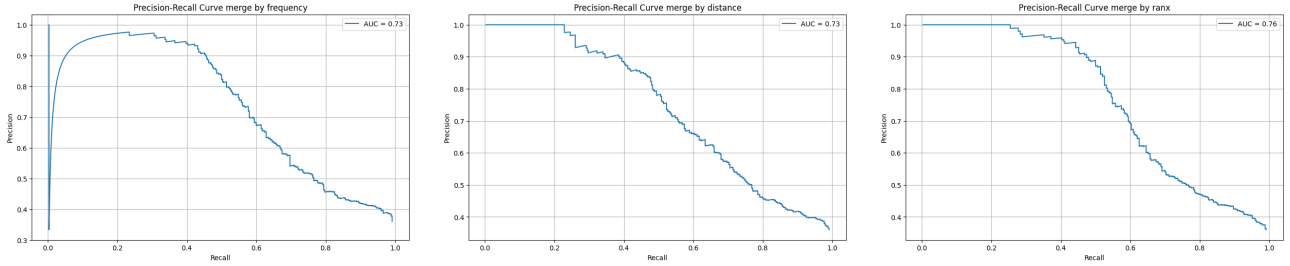

Figure 11: Precision-Recall Curve on fifth try.

- k = 5000, Feature with face: Facenet, **Feature without face**: CLIP (ViT), Remove type: Compare, Merge Result: Yes.

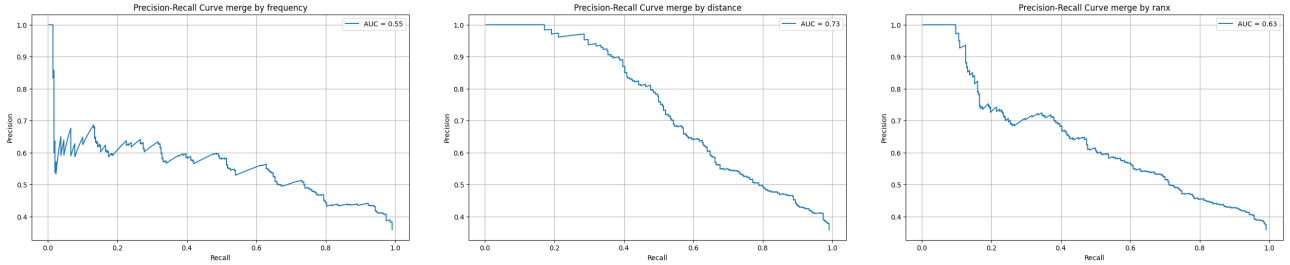

Figure 12: Precision-Recall Curve on sixth try.

- k = 5000, Feature with face: Facenet, **Feature without face**: BEiT (S), Remove type: Compare, Merge Result: Yes.

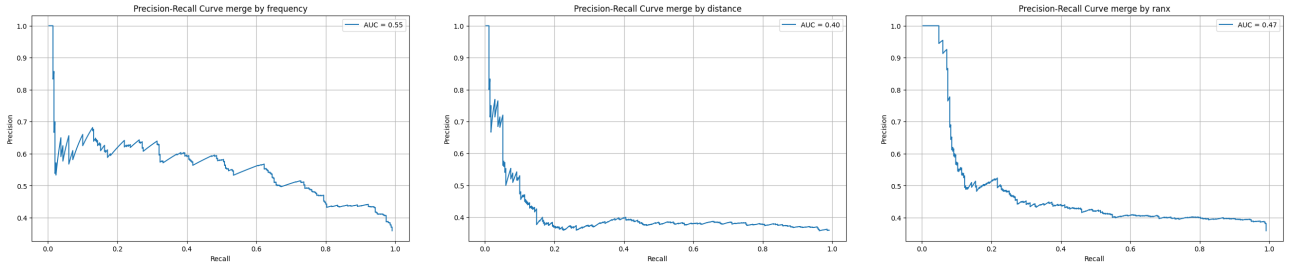

Figure 13: Precision-Recall Curve on seventh try.

- k = 5000, Feature with face: Facenet, **Feature without face**: BEiT (mean), Remove type: Compare, Merge Result: Yes.

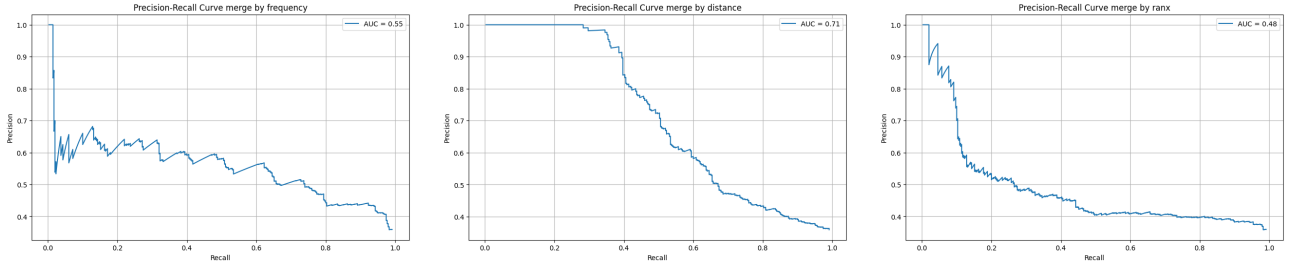

Figure 14: Precision-Recall Curve on eighth try.

After the above process, the selected parameter set is: (k = 5000, Feature with face: Facenet, Feature without face: CLIP (rn), Remove type: Compare, Merge type: distance). However, this is only an evaluation on 1 character; to evaluate objectively, it is necessary to evaluate all characters with MAP. The results are as follows (MAP of the selected set: 0.5920):

Changed step	**Change value**	**MAP**
Feature with face	ArcFace	0.7400
Feature without face	CLIP (ViT)	0.5612
Feature without face	BEiT (S)	0.3918
Feature without face	BEiT (mean)	0.4767
Remove type	DBSCAN	0.6343
Merge type	No	0.3498
Merge type	Frequency	0.4516
Merge type	Ranx	0.5426

Table 2: MAP with all queries for each parameters set

Through the above experiment, it can be seen that the best set with k=5000 is: Feature with face: Facenet, Feature without face: CLIP (rn), Remove type: DBSCAN, Merge type: distance . Here, ArcFace has a high MAP, however, it is due to some problem that makes the face unable to be detected, so the system completely uses the CLIP feature.

Since each character has a different number of sample images, here is a survey of how changing the number of query images of each character affects the results (parameter set selected as above):

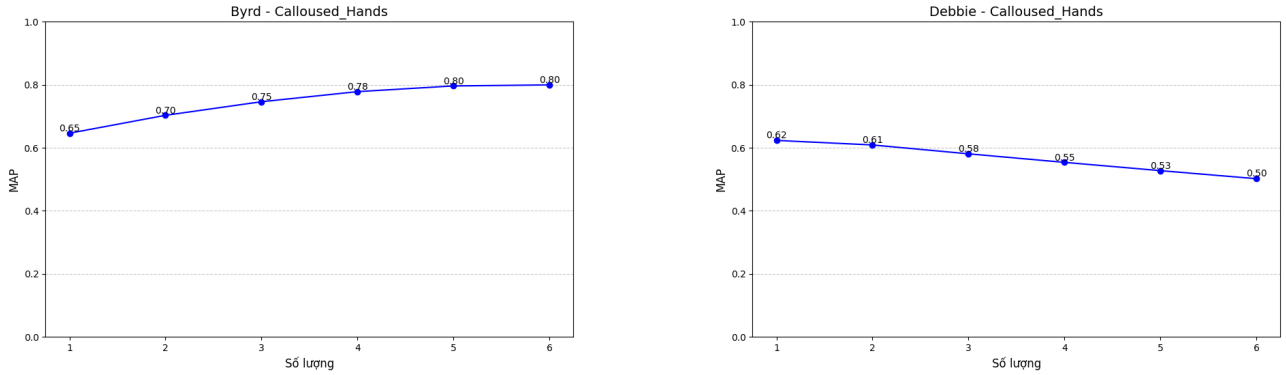

Figure 15: Survey of changes in the number of queries in Caloused Hands

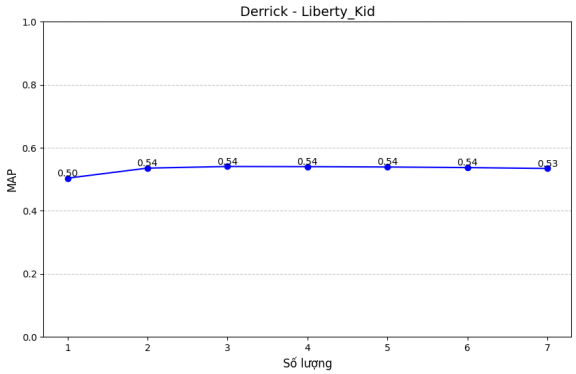

Figure 16: Survey of changes in the number of queries in Liberty Kid

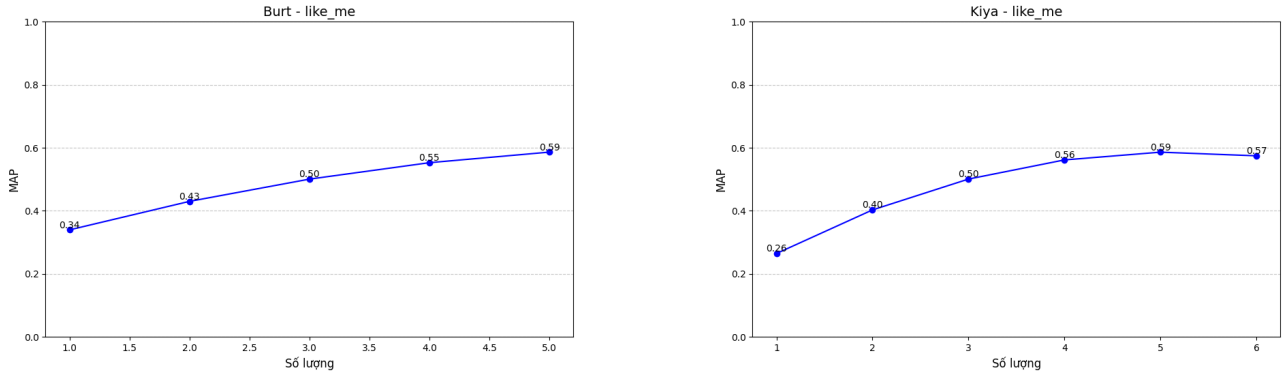

Figure 17: Survey of changes in the number of queries in Like Me

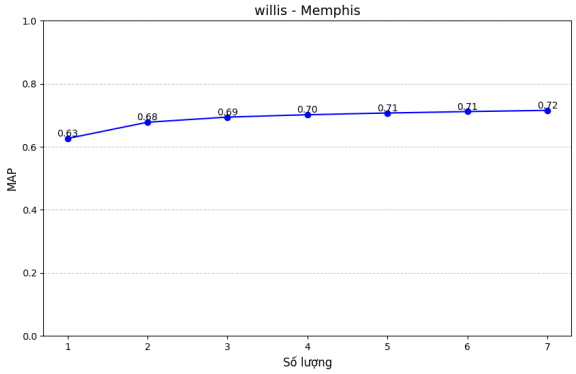

Figure 18: Survey of changes in the number of queries in Memphis

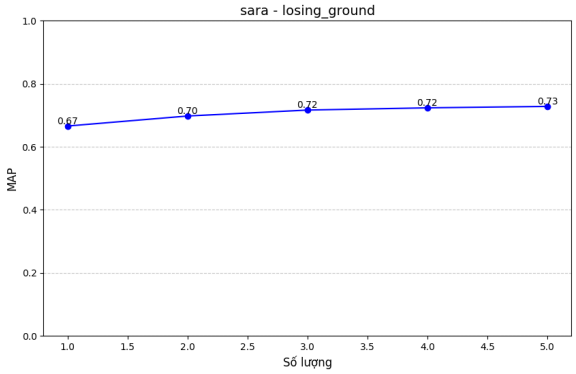

Figure 19: Survey of changes in the number of queries in Losing Ground

5 Reviews

Pros:

- The processing steps are clear from face detection, feature extraction to using FAISS for quick search. This helps maintain transparency and ease of future adjustments.
- The system can scale as the number of videos or queries increases without complicating the process.

Cons:

- Not processed with augmented data.
- Query expansion is not performed yet (because according to the last survey, the more queries, the more effective).
- Feature vectors are taken from models trained on other data, so some may not be appropriate (it's pretty hard to retrain and there's no guarantee it will be better).
- The method still does not achieve 100% recall despite the different implementation from class presentation. It could be caused by removing duplicated.

6 References

Arandjelovic, O. and A. Zisserman (2005). "Automatic face recognition for film character retrieval in feature-length films". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1, 860–867 vol. 1. DOI: 10.1109/CVPR.2005.81.

Arandjelović, Ognjen and Andrew Zisserman (2006). "On Film Character Retrieval in Feature-Length Films". In: *Interactive Video: Algorithms and Technologies*. Ed. by Riad I. Hammoud. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 89–105. ISBN: 978-3-540-33215-2. DOI: 10.1007/978-3-540-33215-2_5. URL: https://doi.org/10.1007/978-3-540-33215-2_5.

Bao, Hangbo et al. (2021). "Beit: Bert pre-training of image transformers". In: *arXiv preprint arXiv:2106.08254*.

Bassani, Elias and Luca Romelli (2022). "ranx.fuse: A Python Library for Metasearch". In: *CIKM*. ACM, pp. 4808–4812. DOI: 10.1145/3511808.3557207.

Cormack, Gordon V., Charles L. A. Clarke, and Stefan Büttcher (2009). "Reciprocal rank fusion outperforms condorcet and individual rank learning methods". In: *SIGIR*. ACM, pp. 758–759.

Deng, Jiankang et al. (June 2019). "ArcFace: Additive Angular Margin Loss for Deep Face Recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Douze, Matthijs et al. (2024). "The Faiss library". In: arXiv: 2401.08281 [cs.LG].

Qiao, Shishi et al. (2020). "Deep Heterogeneous Hashing for Face Video Retrieval". In: *IEEE Transactions on Image Processing* 29, pp. 1299–1312. DOI: 10.1109/TIP.2019.2940683.

Radford, Alec et al. (2021). *Learning Transferable Visual Models From Natural Language Supervision*. arXiv: 2103.00020 [cs.CV]. URL: https://arxiv.org/abs/2103.00020.

Schroff, Florian, Dmitry Kalenichenko, and James Philbin (June 2015). "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Serengil, Sefik and Alper Ozpinar (2024). "A Benchmark of Facial Recognition Pipelines and Co-Usability Performances of Modules". In: *Journal of Information Technologies* 17.2, pp. 95–107. DOI: 10.17671/gazibtd.1399077. URL: https://dergipark.org.tr/en/pub/gazibtd/issue/84331/1399077.

Standards, National Institute of and Technology (NIST) (2022). *TRECVID 2022: Digital Video Understanding (DVU)*. https://www-nlpir.nist.gov/projects/tv2022/dvu.html. Accessed: 2025-01-13.