Assignment #4: Deep Face Recognition Pipeline

Biljana Vitanova
IBB 2024/25 , FRI, UL
bv7063@student.uni-lj.si

***Abstract*—The goal of this assignment was to implement a face recognition pipeline using deep learning approaches. Pretrained models were used for both face detection and recognition steps, which improved performance compared to handcrafted methods.**

I. Introduction

Deep learning models are widely applied across various domains. Trained on large datasets and fine-tuned for specific tasks, they often outperform traditional methods. In face recognition pipelines, they improve face detection by predicting bounding boxes, enhance recognition by capturing more discriminative embeddings, and aid in alignment.

II. Methodology

For the **face detection step**, I experimented with three deep learning-based methods:

1) **YOLOv8**:[1] a **single-pass** model with a variant **fine-tuned** for **face detection**. The model divides the input image into a grid and directly predicts bounding boxes without relying on predefined anchors. Non-Maximum Suppression (NMS) is then applied to remove overlapping boxes and keep the most confident predictions.
2) **MTCNN**:[2] which is designed for face detection and landmark localization, processes each image in multiple steps through cascaded stages. It consists of three CNNs, each refining bounding boxes and removing false positive detections.
3) **RetinaFace(InsightFace)**: uses a backbone network to extract features and a feature pyramid to detect faces at multiple scales.

For the **face recognition step**, I experimented with three deep learning approaches:

1) **ArcFace**: Extracts face features using ResNet and clusters similar embeddings tightly with angular margin loss.
2) **VGGFace**: An older model for face feature extraction, designed for general face recognition tasks.
3) **FaceNet**: Extracts face features by minimizing triplet loss to ensure embeddings of the same person are closer together.

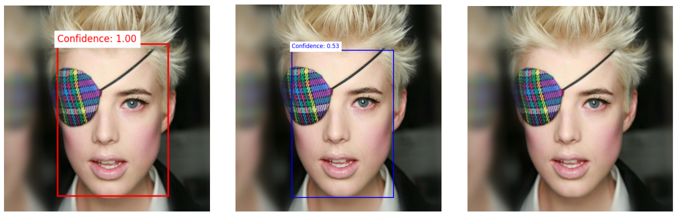
[1]YOLO: You Only Look Once
[2]MTCNN: Multi-task Cascaded Convolutional Networks

III. Experiments

For this assignment, I used the CelebA-HQ-small dataset [1], which consists of 475 training images and 412 test images, each of size 512×512 pixels.

All the models were pretrained on Face Datasets. Since all the models were pretrained, on there was no need to further fine-tune the parameters.

For the **face detection step**, there were no **false positive detections** of faces, as each image resulted in at most one face detection bounding box. The MTCNN model predicted faces with **confidence greater than 0.9** in all cases. For the other two models, the confidence of the bounding boxes varied between **0.3 and 0.7**. However, there were instances where RetinaFace and YOLOVv8 **failed** to estimate a face due to **occlusion**, which was not the case for MTCNN.



(a) MTCNN (b) YOLOv8 (c) RetinaFace

Fig. 1: Face detection and confidence comparison for the three face detection models.

For the **face recognition step**, embeddings were extracted from the **second last layer**. For FaceNet, the model used was InceptionResnetV1, pretrained on the VGGFace2 dataset. Each image was resized to 160x160 and standardized, resulting in a 512-dimensional embedding after processing. For ArcFace, the face recognition output was 512-dimensional, and for VGGFace, the output was 4096-dimensional, using DeepFace library [2] for both models, there was no need for prior processing. All the embeddings were subsequently **normalized**.

For the **face matching step**, using Hellinger distance resulted in **suboptimal results**. As stated in [3], all feature extractors were trained to **minimize cosine** or **Euclidean distance.**

IV. Results and Discussion

The performance of the recognition pipeline was evaluated separately for the detection step, the feature extraction step, and the entire pipeline as a whole.

A. Results

TABLE I: Mean IoU and execution time for different deep learning-based face detection methods.

Model	Mean IoU[%]	Execution Time[s]
YOLOv8	88.92	**42.06**
MTCNN	**99.82**	69.47
RetinaFace	89.69	63.31
Viola Jones	71.68	291.85

TABLE II: Rank-1 and Rank-5 on whole images

Model	Rank-1 Accuracy[%]	Rank-5 Accuracy[s]
FaceNet	**92.23**	**97.57**
VGGFace	84.46	89.80
ArcFace	82.28	84.70

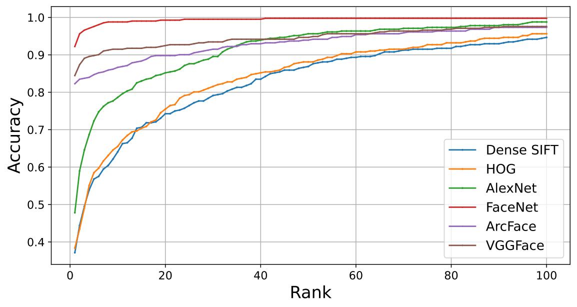


Fig. 2: **Cumulative Match Characteristic on whole images** (CMC) curve showing the retrieval accuracy for each descriptor, with accuracy displayed for ranks up to 100.

TABLE III: Rank-1 and Rank-5 on complete face recognition pipeline

Model	Mean IoU[%]	Execution Time[s]
FaceNet	**98.05**	**99.75**
VGGFace	84.70	91.26
ArcFace	89.56	92.71

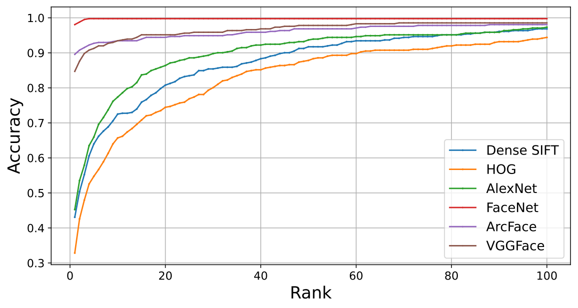


Fig. 3: **Cumulative Match Characteristic (CMC)** curve showing the retrieval accuracy of the complete face recognition pipeline for each descriptor, with accuracy displayed for ranks up to 100

B. Discussion

YOLOv8, as a single-pass model, had the shortest execution time but slightly lower IoU. In contrast, MTCNN, with its multi-stage design, achieved the highest IoU of 99.82% but required more time to execute. Despite these differences, all three deep learning methods outperformed the handcrafted Viola Jones method.

For face recognition, the FaceNet feature extractor had the best rank accuracy, followed by VGGFace and ArcFace. Using the MTCNN face detector improved the overall recognition process for FaceNet and VGGFace but did not improve rank accuracy for ArcFace. However, all the methods outperformed simpler approaches such as HOG, SIFT, or the AlexNet CNN, which was not trained on face images.

V. Conclusion

This assignment demonstrated the effectiveness of incorporating deep learning-based methods in face recognition pipelines, particularly those fine-tuned for performing recognition and detection tasks.

For future work, it would be valuable to experiment with different feature extraction methods, compare their performance, and evaluate the importance of the alignment step, landmark extraction, and other preprocessing techniques.

References

[1] W. Xia, Y. Yang, J.-H. Xue, and B. Wu, "Towards open-world text-guided face image generation and manipulation," *arxiv preprint arxiv: 2104.08910*, 2021.

[2] S. Serengil and A. Ozpinar, "A benchmark of facial recognition pipelines and co-usability performances of modules," *Journal of Information Technologies*, vol. 17, no. 2, pp. 95–107, 2024. [Online]. Available: https://dergipark.org.tr/en/pub/gazibtd/issue/84331/1399077

[3] S. I. Serengil. (2020, December) Deep face recognition with arcface in keras and python. [Online]. Available: https://sefiks.com/2020/12/14/deep-face-recognition-with-arcface-in-keras-and-python/