Face Verification with VGGFace and MTCNN

Robin Tran

***Abstract*— The main objective of this project is to enhance the accuracy of face verification, determining if two face images represent the same individual, by combining VGGFace embeddings with Multi-task Cascaded Convolutional Networks (MTCNN) for face detection. Face verification is an important part of computer vision, and is used in various applications, such as identity verification, security, and social media tagging. While VGGFace is widely used for face recognition, it might face challenges in accurately detecting faces in images with complex and noisy backgrounds, which can lead to errors in identity verification. In this project, we use MTCNN as a preprocessing step to accurately detect faces from the images, then apply VGGFace to verify if two images are of the same person. We use the Labeled Faces in the Wild (LFW) dataset for testing, and assess the improvement in verification accuracy by introducing MTCNN.**

***Index Terms*— Face recognition; Multi-task Cascaded Convolutional Networks; VGGFace**

I. Introduction

Face verification is an important area within face recognition, where the aim is to confirm whether two face images belong to the same person. In verification tasks, a model generates embeddings, or vector representations, of the two images. If the similarity score between embeddings is above a certain threshold, the faces are considered the same [1].

VGGFace is a deep CNN model designed mainly for face recognition tasks. It was developed by University of Oxford. It produces high-dimensional embeddings that represent unique facial features for similarity comparisons [2].

The **Multi-task Cascaded Convolutional Networks (MTCNN)** model, introduced by Zhang et al., performs face detection and alignment simultaneously [3]. This method is able to detect faces with high accuracy even in complex and clustered environments.

This project aims to explore existing approaches to the current challenge, and specifically, assess whether adding the Multi-task Cascaded Convolutional Networks (MTCNN) as a preprocessing step enhances the accuracy of face verification by VGGFace.

II. Existing Methods and Open Challenges

Several deep learning models have been developed to address the face verification task, with some of the most popular: **FaceNet**, **VGGFace**, and **Siamese Networks** [2] [4]. While these models achieve high accuracy, they might still face challenges, particularly in handling images with cluttered or complex backgrounds, which can affect detection accuracy.

To address these challenges, researchers have experimented with various methods aimed at improving face detection and verification in complex scenes.

A. Skin Color Segmentation

One approach for finding faces in complex backgrounds is skin color segmentation. This method uses color to identify areas in an image that might be skin, then narrowing down potential face regions. By focusing on these areas, the model reduces the search space, which makes locating faces easier [5] However, this approach can be affected by lighting and skin tones. For example, skin color can look different in the natural sunlight and inside light. Although this method is not perfect, skin color segmentation can be a useful first step when combined with other methods to improve face detection.

B. AdaBoost

AdaBoost is an algorithm that combines several simple classifiers to create a stronger one, which can detect faces even in busy backgrounds. It works by selecting the most important features, like edges or textures, that help separate faces from the background. By focusing on only the most useful features, AdaBoost makes face detection faster and more accurate. This method is often used in real-time applications and is especially helpful in images with a lot of background noise, as it narrows down the search area for faces [6]. A limitation of AdaBoost is that it can be sensitive to noisy data or outliers, sometimes leading to false positives. It can also become computationally expensive as it adds classifiers, which may slow down performance in real-time applications [7].

C. Multi-task Cascaded Convolutional Networks

Multi-task Cascaded Convolutional Networks (MTCNN) is a tool that helps with both finding and aligning faces in images, all in one model. It works in three steps. First, it quickly looks for areas that might contain a face. Second, it refines or checks these areas to confirm they're actually faces. Third, it locates key points on the face, like the eyes and nose, to make sure the face is aligned properly. This step-by-step process helps MTCNN accurately detect faces, even in busy or crowded images, and aligns them well [3].

D. Research Idea

Our experiments showed that MTCNN effectively detects and aligns faces, so we want to test if using MTCNN as a preprocessing step can improve VGGFace's accuracy in face verification.

Specifically, we aim to see if MTCNN's role in face detection and alignment, giving VGGFace consistently cropped and centered images, leads to more reliable and accurate embeddings for identity verification. By standardizing the

input, we hypothesize that MTCNN preprocessing will produce higher-quality embeddings from VGGFace, improving its ability to verify faces accurately.

To measure the impact of MTCNN preprocessing on VGGFace's accuracy, we will use cosine similarity. This metric checks how similar two embeddings are by calculating the cosine of the angle between them, showing how closely the vector representations match [8].

Cosine similarity, $\cos(\theta)$, is defined as:

$$\text{Cosine Similarity} = \cos(\theta) = \frac{A \cdot B}{||A|| \times ||B||}$$

where A and B are the embeddings of two images, $A \cdot B$ denotes their dot product, and $||A||$ and $||B||$ are the magnitudes of the embeddings. A score close to 1 suggests high similarity (same person), while a score closer to 0 implies lower similarity (different people) [8].

By comparing accuracy of the prediction (based on cosine similarity scores) before and after introducing MTCNN as a preprocessing step, we will assess whether MTCNN preprocessing improves VGGFace's performance in face verification.

III. Concept to Code

In this section, we describe the dataset used and outline the technical steps in the implementation. The code source can be found here https://github.com/MHC-FA24-CS341CV/beyond-the-pixels-emerging-computer-vision-research-topics-fa24/blob/main/code/05-face-recognition/05facerecognition.ipynb.

A. Dataset Description

The **Labeled Faces in the Wild (LFW)** dataset is used to evaluate the face verification accuracy of the proposed approach. LFW is a very popular dataset for face verification tasks, containing over 13,000 labeled images of faces in diverse settings. This dataset challenges the models with variations in background, lighting, and pose, so it is helpful for testing our MTCNN and VGGFace-based pipeline. The dataset is publicly available from the University of Massachusetts Amherst and is downloaded and extracted for testing [9]. We prepare the LFW dataset as image pairs labeled "same" or "different," enabling straightforward comparison of similarity scores for verification.

B. Environment Setup and Run Instructions

To set up and run the code, ensure the following packages are installed in your environment [10]:

- **TensorFlow and Keras**: For building and running the VGGFace model.
- **MTCNN**: For face detection and alignment.
- **NumPy**: For array manipulation and cosine similarity calculations.
- **Matplotlib**: For visualizing face detections and verification results.

We also download the pretrained VGGFace weights from Kaggle, which are used to initialize the VGGFace model and enable it to produce embeddings without having to train again [11].

C. Experiments with MCTNN

We experimented with MCTNN on the LFW dataset, and observed that it can correctly detect and align faces.

Fig. 1: MTCNN Face Detection and Alignment

D. Pipeline Implementation

The pipeline combines MTCNN for face detection and alignment with VGGFace for embedding generation, aiming to maximize the model's face verification accuracy by providing high-quality inputs.

1) Image Processing:

- **With MTCNN**: Each image is processed through MTCNN, which detects and aligns faces by locating key landmarks (like eyes and nose) and centering each face in the frame. This aligned image is then resized to 224x224 pixels for compatibility with VGGFace.
- **Without MTCNN**: The image is directly resized to 224x224 pixels without any face detection or alignment. This serves as a baseline, allowing us to measure the effect of MTCNN preprocessing.

2) Embeddings: Both MTCNN-preprocessed and baseline images are passed through VGGFace, which generates a high-dimensional embedding for each image. These embeddings represent the face features and are used to determine the similarity between pairs of images [1].

3) Cosine Similarity: For each pair of embeddings, we calculate cosine similarity, which measures how close the embeddings are. A higher cosine similarity score suggests that the two images are likely of the same person, while a lower score suggests different individuals. We define the

threshold to be 0.5, so if a pair's cosine similarity is higher than 0.5, the images belong to the same person.

4) Performance Metrics: We evaluate verification accuracy using metrics such as accuracy, precision, recall, and F1 score, comparing VGGFace's performance with and without MTCNN preprocessing. This comparison helps determine if MTCNN preprocessing consistently improves verification accuracy.

E. Model Architecture

The VGGFace model is built to create embeddings that capture the key features of each face. It starts with convolutional layers with filter sizes increasing from 64 to 512, which help pick up on facial details. After each set of convolutions, there are pooling layers that reduce the image size to focus on the most important features and make the model faster. Then, fully connected layers with 4096 filters each combine the features into high-dimensional embeddings. Dropout is used here to help avoid overfitting. Finally, the embedding layer gives a 2622-dimensional vector that represents each face [12].

IV. Output and Results

This section presents the results of the face verification model, comparing the outcomes with and without MTCNN preprocessing.

Firstly, Figure 2 displays example outputs of the model, showing original images alongside their MTCNN-processed counterparts. These figures illustrate how MTCNN preprocessing aligns and centers the faces, which are then fed into the VGGFace model for embedding generation.

We also obtain the performance metrics for face verification with and without MTCNN preprocessing. Table I compares accuracy, precision, recall, and F1 score between the two setups.

Metric	With MTCNN	Without MTCNN
Accuracy	0.558	0.568
Precision	1.000	1.000
Recall	0.116	0.136
F1 Score	0.208	0.239

TABLE I: VGGFace performance with and without MTCNN

The results indicate that, opposite from our hypothesis, adding MTCNN preprocessing had little impact on VGGFace's overall verification accuracy. While MTCNN preprocessing slightly improved precision, making the model more skeptical in identifying matches, it also led to a higher rate of missed same-person pairs. Although the current setup didn't yield significant improvements, we believe that MTCNN preprocessing shows potential and could be explored further to enhance existing deep learning models' performance in addressing difficult face verification tasks.

V. Next Steps

We can compare VGGFace's performance with other state-of-the-art face verification models (such as FaceNet), to assess whether a better solution is possible.

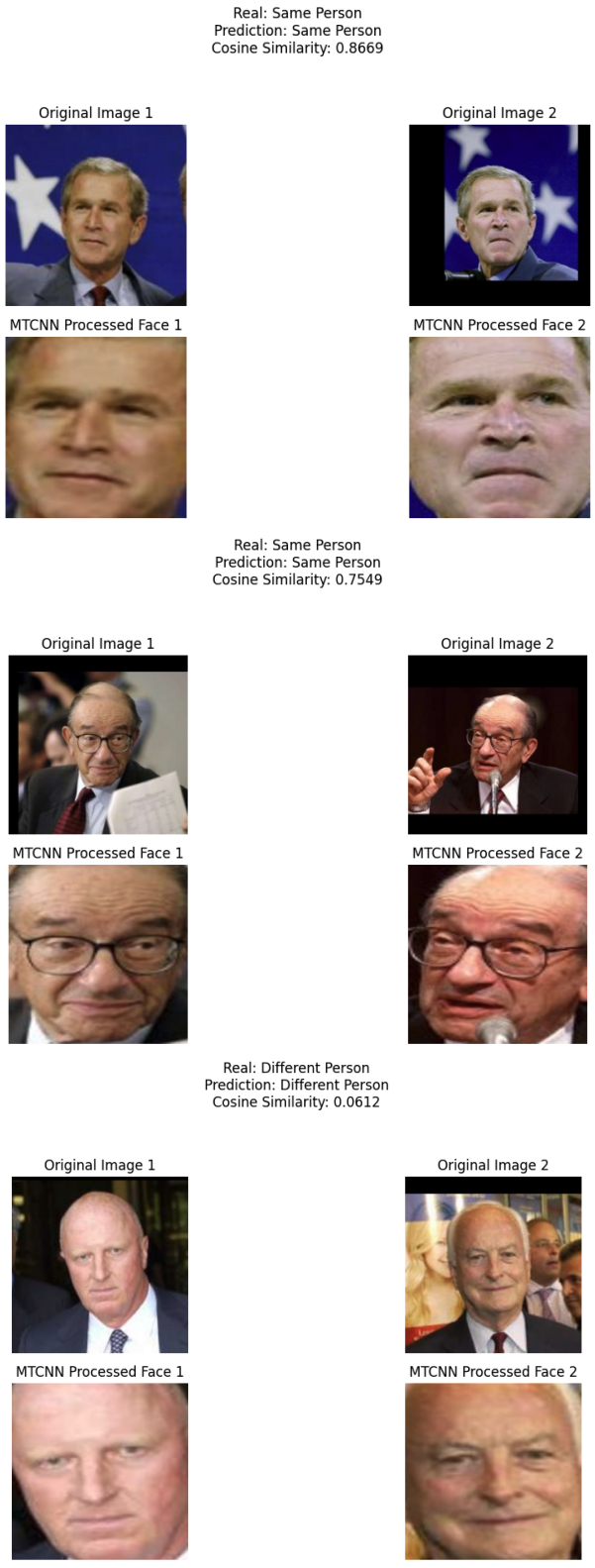

Fig. 2: Examples of original and MTCNN-processed faces used for verification.

References

[1] Y. Huang, J. Xu, and S. Ding, "Face feature embedding," in *Handbook of Face Recognition*, S. Z. Li, A. K. Jain, and J. Deng,

Eds. Springer, Cham, 2024, published 30 December 2023. [Online]. Available: https://doi.org/10.1007/978-3-031-43567-6$_8$

[2] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2015.

[3] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multi-task cascaded convolutional networks," *IEEE Signal Processing Letters*, 2016.

[4] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *arXiv preprint arXiv:1503.03832*, 2015, arXiv:1503.03832 [cs.CV]. [Online]. Available: https://doi.org/10.48550/arXiv.1503.03832

[5] Y.-T. Pai, S.-J. Ruan, M.-C. Shie, and Y.-C. Liu, "A simple and accurate color face detection algorithm in complex background," †E-mail: sjruan@mail.ntust.edu.tw.

[6] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004. [Online]. Available: https://doi.org/10.1023/B:VISI.0000013087.49260.fb

[7] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139–157, 2000. [Online]. Available: https://doi.org/10.1023/A:1007607513941

[8] H. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," in *Computer Vision – ACCV 2010*, ser. Lecture Notes in Computer Science, R. Kimmel, R. Klette, and A. Sugimoto, Eds., vol. 6493. Springer, Berlin, Heidelberg, 2011, pp. 709–720. [Online]. Available: https://doi.org/10.1007/978-3-642-19309-5$_5$5

[9] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007. [Online]. Available: http://vis-www.cs.umass.edu/lfw/lfw.pdf

[10] K. N. We utilized several Python libraries for this project, including TensorFlow and Matplotlib.

[11] R. Acharya, "Vgg face model weights for keras models," updated 4 years ago, available at https://www.kaggle.com.

[12] V. Pillai, "Facial recognition using vggface keras," updated 2 years ago, available at https://www.kaggle.com.