Evaluation of InsightFace Face Recognition Models on LFW, CALFW, and CPLFW Datasets Using Client-Server Setup

Kourosh Hassanzadeh

In this project, I was tasked with calculating the accuracy of face recognition models from the InsightFace repository on specified datasets and reporting the results. For consistency and standardization, I used the pre-trained models **buffalo_s** and **buffalo_l** available in the repository. The accuracy of the models on the LFW, CPLFW (LFW-CP), and CALFW (LFW-CA) datasets was evaluated, with a focus on calculating both the FMR (False Match Rate) and FNMR (False Non-Match Rate) for each model on each dataset. The accuracy was computed on 6,000 pre-defined 1:1 pairs from the dataset files.

To implement the solution, I developed an API using flask-restful, which facilitated communication between the client and server. The client sent requests with pairs of images, and the server, using the pre-trained models, calculated the similarity and returned the results. The server handled model loading and inference, while the client sequentially sent 6,000 requests and computed the overall accuracy based on the server's responses.

1. Datasets:

1.1. LFW (Labeled Faces in the Wild): This dataset contains around 13,000 face images of individuals, which can be extracted along with their labels from the pairs.txt file. The images have dimensions of 250 × 250 × 3 pixels. The text file includes pairs of images, and each entry indicates whether the two images match or not. The text file has two cases:

1.1.1. Abel_Pacheco 1 4: This means that images 1 and 4 from the folder Abel_Pacheco are the same.

1.1.2. Phil_Morris 1 Yuri_Luzhkov 1: This means that the image 1 from Phil_Morris and the image 1 from Yuri_Luzhkov are not the same.

Figure 1 - A matching pair

Figure 2 - A non-matching pair

1.2. CPLFW (Cross-Pose LFW): This dataset contains 11,652 images of individuals in different poses, which can be extracted along with their labels from the pairs.txt file. The images have dimensions of 224 × 224 × 3. The text file should be read two lines at a time, as shown below:

- Ingrid_Betancourt_1.jpg 1
- Ingrid_Betancourt_2.jpg 1

This means that the specified images are a match. If the label is 0, it means the images are not a match.

Figure 3 - A matching pair

Figure 4 - A non-matching pair

1.3. CALFW (Cross-Age LFW): This dataset contains 12,174 images of individuals at different ages, which can be extracted along with their labels from the pairs.txt file. The images have dimensions of 224 × 224 × 3. The text file should again be read two lines at a time, with the difference that labels greater than zero indicate that the images are a match, and if the label is zero, they are not a match.

- Steve_Karsay_0001.jpg 6
- Steve_Karsay_0003.jpg 6

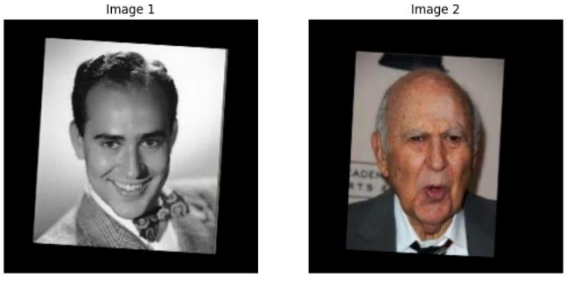

Figure 5 - A matching pair

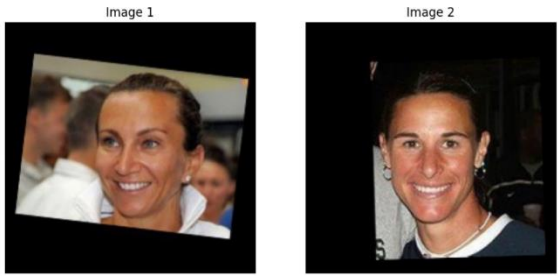

Figure 6 - A non-matching pair

- ❖ These datasets are on the client side, and the server only receives the images as pairs in sequence.

2. Client-Side Processing:

   Initially, the client extracts image pairs and their labels from the pairs.txt file and stores them in an array. The images are then loaded with the load_image() function, encoded, and prepared for sending to the server.

3. Sending Data to the Server:

   Using the **requests** library, the client sends a POST request to the server at 'http://127.0.0.1:5000/testModel', along with the two images and the selected model in a JSON format.

4. Server-Side Processing:

   On the server side, using flask, the server receives the request and processes it through the TestModel class with the following functionality:

   4.1. The mentioned models were **buffalo_s** and **buffalo_l**, with buffalo_l being heavier and more complex, resulting in better accuracy for the buffalo_l model. In this class, both models are loaded, and depending on the model sent from the client, a final model is selected for testing. The det_size parameter was adjusted through trial and error to be 480 × 480.

   4.2. In the `post` method, which handles requests sent from the client, the **load_image_from_base64()** function is first used to decode two images sent from the client, and the model sent is also loaded into the class model. Additionally, if the model is unknown, an error Response is returned.

   4.3. Then, using the loaded model, we extract the faces and embedding vectors of both images and measure the similarity between the two images using **cosine similarity**. We then send a response to the client indicating whether the two images are identical or not. If their similarity is greater than **0.2**, the match field is set to True, and if it is less, it is set to False. This result, along with the status, is sent to the client. Additionally, if the face model does not detect any faces, a missMatch result of False is returned.

```
faces1 = model.get(img1)
faces2 = model.get(img2)

if len(faces1) > 0 and len(faces2) > 0:
    embedding1 = faces1[0].embedding
    embedding2 = faces2[0].embedding
```

Figure 7 - How to get output from the model

4.4. The threshold was selected using greed search and research. I found that most projects set this threshold at 0.27. By using a greedy search, I arrived at a threshold of 0.2. Increasing the threshold reduces the model's accuracy but decreases FMR (False Match Rate) and increases FNMR (False Non-Match Rate). Conversely, lowering the threshold improves accuracy but increases FMR and decreases FNMR. Thus, a trade-off must be established, and a threshold of 0.2 was found to be the optimal balance.

5. Client-Side Results:

5.1. If the images are a match and the model correctly identifies them, the result is True (correct).

5.2. If the images are not a match and the model correctly identifies them, the result is False (correct).

5.3. If the images are a match but the model fails to identify them, the result is False (FNMR).

5.4. If the images are not a match but the model falsely identifies them as such, the result is True (FMR).

6. Accuracy and Results:

Dataset	Dataset	Accuracy	FMR	FMNR
lfw	buffalo_s	98.15%	14	97
lfw	buffalo_l	98.58%	7	78
cplfw	buffalo_s	90.45%	127	446
cplfw	buffalo_l	94.28%	71	272
calfw	buffalo_s	94.13%	101	251
calfw	buffalo_l	95.52%	48	221

Table 1 – Results Obtained