# Chapter 9

# Face Recognition – The `compare_faces` method

Let us recall the list of features we have learned so far in **`face_recognition`** library. After learning to load an image as a Python NumPy object using **`load_image_file()`** method, we have studied the following key methods:

1. *Face Detection*: The **`face_locations()`** method: Given an image as a Python NumPy object, this method scans the entire image, pixel by pixel, detects the rectangular segments of human faces in the picture, and returns the Python list of all the human faces. Here each face is represented as a rectangular frame with pixel numbers (top, right, bottom, left).
2. *Face Encoding / Face Signature*: The **`face_encodings()`** method: Given an image as a Python NumPy object, this method scans the entire image, pixel by pixel, and detects the rectangular segments of each of the human face in the picture. For each of the rectangular face, it computes the embedding (signature) of the face in the form of a 128-vector. If there are *n* faces in the image, the method computes the encodings of each of the *n* face and returns a Python list of *n* signatures.
3. Face Distance: The **`face_distance()`** method: Given one face encoding **`fx`** and a list of n face encodings **`faces[]`**, the **`face_distance(faces, fx)`** method computes the distance of each member faces of faces[] to fx as a real number between 0 and 1 and returns the Python list of distances **`d[]`**.

In this chapter, we introduce one more method **`compare_faces()`**, that compares one face **`fx`** with a list of faces **`faces[]`** and returns a Boolean list that tells what are the faces that matches with fx. If **`faces[]`** has n faces, then method call

```
compare_faces(faces,fx)
```

returns a Boolean array **`match[]`**, where,

$$\mathbf{match[i]} = \begin{cases} True\ if\ faces\ \mathbf{fX}\ and\ \mathbf{faces[0]}\ match \\ False\ Otherwise. \end{cases}$$

In the package **face_recognition**, the method, **compare_ faces()** is used to compare faces and recognize the faces. Let us first see a very simple program.

### 9.2 Compare one face with a list of faces

Now, we can check this entry and decide in the unknown match with known dace (Xavier) The method is described below:

Consider a list of face encodings of *n* pictures.

$$faces = [f_0, f_1, f_2, f_3, \ldots f_n].$$

We want to compare each face encoding $f_i$ with another unknown face encoding *f*. That is, we want to find the values of

$$match_i = \begin{cases} True\ if\ f_i and\ f\ match \\ False\ Otherwise. \end{cases}$$

In Python **face_recognition** package, we write as follows:

```
faces = [f0, f1, f2, f3, …fn]
match = face_recognition.compare_faces(faces, fX)
```

The result **`match`** is a Boolean list **`match[i] = True or False`** which tells if **`faces[i]`** matches with **`fX.`**

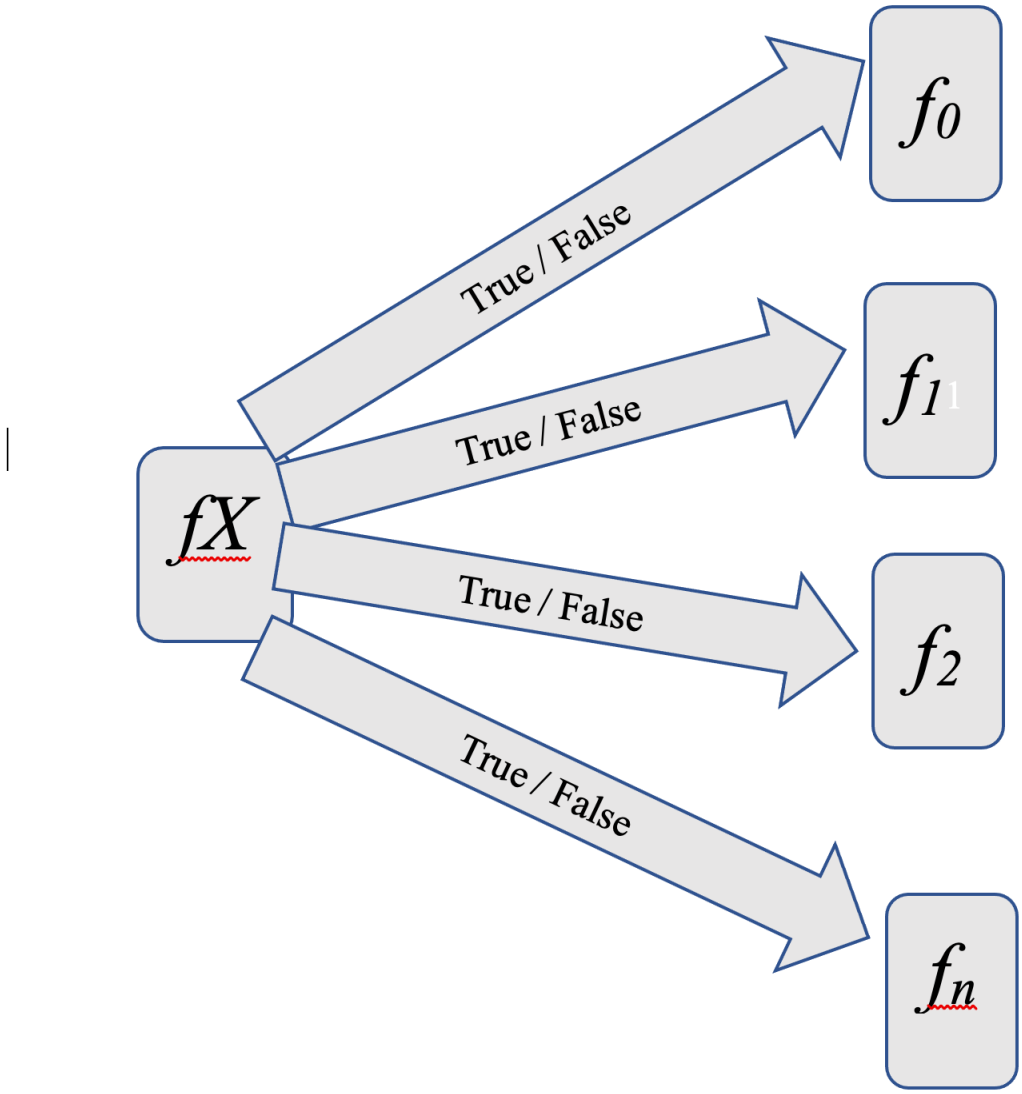


**Figure 9.3 The `face_compare()` method**