SoC Summer Workshop

Project

Masked Unmasked Face Recognition

Figure 1: Faces wearing masks.

Face Recognition is a popular biometric authentication method today. As people are wearing masks due the Covid-19 pandemic, it is hard for the face recognition methods to function as intended since the face is now only partially visible. This poses a problem in many scenarios where face recognition is used regularly. For example, at a border control when people cross from one country to another.

In this project, you are required to build the following:

1. **Create a masked face dataset**: Due to the lack of public dataset with masked faces, you will create a synthetic dataset from an existing unmasked face dataset.
2. **A Facemask detector**: Given a face image, this will detect if the person in the image is wearing a mask or not.
3. **Face recognition model for unmasked faces**: This face recognition model will authenticate people who are NOT wearing any mask.
4. **Face recognition model for masked faces**: This model will authenticate people who may be wearing masks. That is, the

This project is a great opportunity to learn the basics of face recognition and how to tackle the new challenge caused by masks.

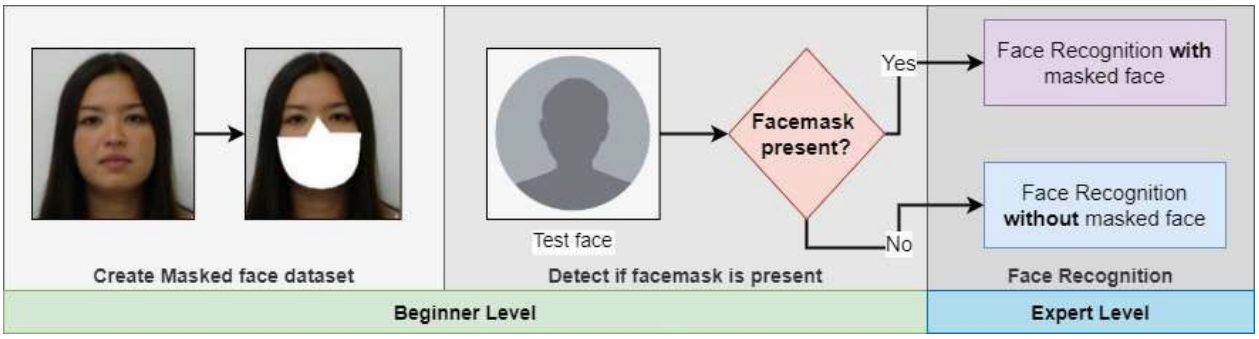

Figure 2: Overview of Beginner and Expert levels.

Beginner Level:

We will use the Georgia Tech Face database which contains images of 50 people taken at a resolution of 640x480 pixels. The average size of the faces in these images is 150x150 pixels. Images of each subject (ie. person) reside in separate folders named "sXX" where "XX" is the number of the subject (from "01" to "50"). Each such folder contains 15 images of a subject, named from "01.jpg" to "15.jpg".

The dataset is available as a zip file in your project folder under the name "Dataset_1.zip".

For the beginner level, we make use of the "starter.py" code, which you can find in the project directory. Follow along with the tasks and fill in the blanks of the given code to complete beginner level.

Tasks:

T1: Reading images.

- Change the dataset_path to point to the unzipped FaceDataset_1/ folder in your computer.
- The given outer loop will go through all the sub-folders in the folder. The inner loop will go through all the images of a sub-folder (each sub-folder belongs to one subject).
- Complete the code to read the images.
- The labels for each image have been already appended to list y for you.

T2: Pre-processing images.

- In the given loops (outer and inner), use the dlib's face shape detector library (sample) with the pre-trained model (download) to locate facial landmarks in an image. This will help you to crop the image in the next step.
- Use the detected landmarks from the previous step to crop the face region out of the image. Now resize the image to 150x150 pixels.
- Convert the cropped and resized image from the previous step to grayscale.
- Complete the rest of the code in starter.py to finish this task.

Figure 3: The steps for creating a digital mask.

T3: Create a masked face dataset.

- Like the previous task, for each image, detect facial landmarks in the image.
- Use an appropriate subset of the landmarks to draw a mask on the image using cv2.fillPoly method.
- After the end of this task, `X_masked` will contain all the images with added digital/synthetic mask.
- Refer to Figure 3.

T4: Mask detector.

- Prepare features using the given code (already done for you).
- Write code to split `X_processed, X_masked` and `y` into training and testing sets. Make use of the *"`sklearn.model_selection.train_test_split`"* to do this. Use an 80-20 split and make sure to shuffle the samples.
- Use the sklearn SVM package to train a binary classifier using `x_train` and `y_train`. The classifier should output if a given image has facemask, or not.
- Use the `x_test` and `y_test` to evaluate the classifier and print the accuracy value.

Expert Level:

We will build upon the beginner level code to try out different techniques and improve our model. The same dataset will be used here.

Complete the following tasks:

Tasks:

T1: Different pre-processing techniques

- What are other pre-processing steps you can use?
- Examples: Perform gaussian blur to reduce noise; or convert to a different color space.
- Try a few other pre-processing techniques and evaluate how they affect accuracy.

T2: Train a Face Recognition model for faces WITHOUT mask.

- Split the 50 subjects into two mutually exclusive sets: 30 subjects into a set called "Employee", and 20 subjects into a set called "Outsider". Think of the 30 subjects as employees authorized to enter an office building which is protected by a face recognition system. The "Outsider" set are non-employees who may try to enter the building.
- Train a face recognizer only on the Employee set; do not use the Outsider set for training. Use

the X_processed (not X_masked) list as X_processed contains only the images WITHOUT facemask.

- The face recognizer should take a probe image (which may be Employee or Outsider) as an input, output "ACCEPT" when the probe image is from the Employee set, and output "REJECTED" when the it is from the Outsider set.
- For a start you can use this face_recognition package. (https://pypi.org/project/face-recognition/)
- Report on the accuracy of the model.

T3: Train a Face Recognition model for faces WITH mask.

- This task is similar to T2. The only difference is that the training and probe images will be faces WITH masks (use the X_masked list this time).
- The other details are same as in T2.

T4: Combine the pipeline with the tasks from Beginner level.

- Combine these tasks to complete the pipeline as shown in Figure 2.

Bonus Level:

In this level we will create a face recognition model (Masked Unmasked Face Matching - MUFM) which can recognize a probe face image of a known subject WITH mask, even when the model was trained using only face images WITHOUT facemasks. See Figure 4 to get a better idea.

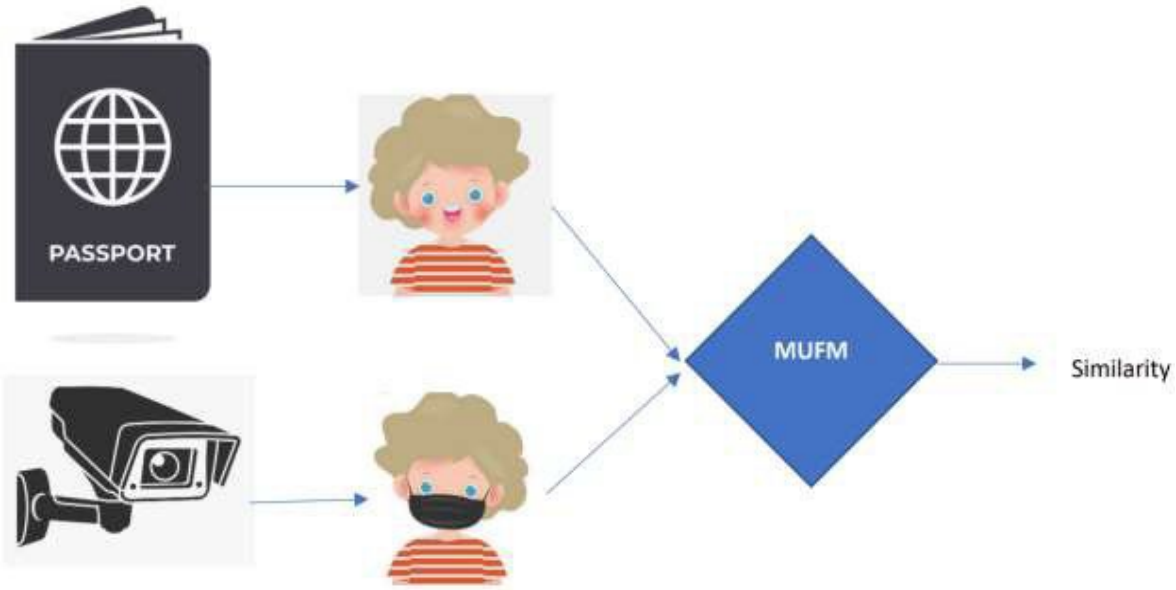

Figure 4: Masked-unmasked face recognition.

First, use the same dataset (Georgia Tech Face database) as in the previous levels to train and test your model. Be careful to use only unmasked face images for training. The probe image, on the other hand, must be with mask. Report on the model accuracy.

Next, use the larger celebA face dataset, which may be obtained as follows:

1. Download the zip file from here (1.34Gb) : https://tinyurl.com/mufm-celebA
2. Download annotations `identity_CelebA.txt` from https://tinyurl.com/mufm-identity
3. Place `identity_CelebA.txt` under `celeba/`

Using the celebA dataset, create synthetic masks as needed. Then train a new model for MUFM using only unmasked images, and test it using probe images that must have masks. Report on the accuracy.

Important Caveats:

1. You must complete Beginner Level. If you don't, you will get 0 for the project. The other levels are optional.
2. For the Expert Level, you may only use the recommended face recognition package, or train your own recognizer using the models provided in scikit-learn.
3. If you use any pre-trained model from the internet (eg ResNet, VGG, ArcFace), or any deep learning network, you may only do this for the Bonus Level.