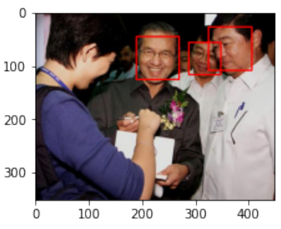
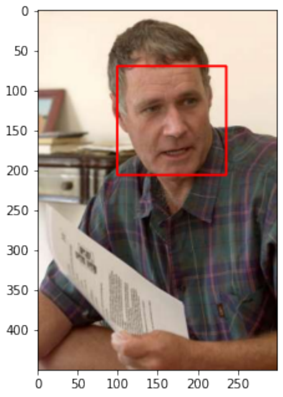
CSE 573 Computer Vision & Image Processing

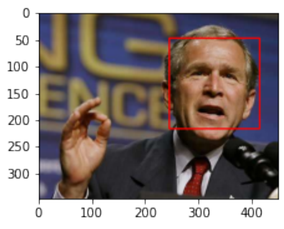
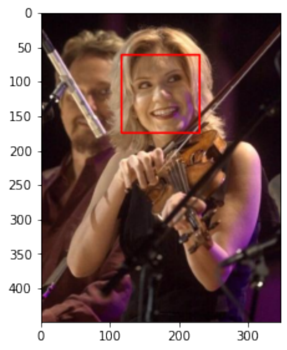
Digvijay Patil (50428152)

Project 3: Face Detection & K-Means Algorithm

- **Algorithms:**
  - Viola Jones Face Detection Algorithm.
  - K-Means Clustering Algorithm.

- **Part A: Procedure & Challenges:**
  1. Initially tried to implement the Viola Jones Face Detection Algorithm by referring to the paper: "Rapid Object Detection using a Boosted Cascade of Simple Features". Progressively implemented parts of the algorithm such as Haar Features of which there were 5 types: Top White Bottom Black, Left White Right Black, Top and Bottom White Middle Black, Left and Right White Middle Black and Chessboard. It could successfully detect the position of features on a sample image. Integral Image and the Intensities within the bounding regions also calculated.
  2. The problem with the above plan was the implementation of Adaboost Classifier which proved to be quite difficult and therefore took the low risk approaching of using the built in OpenCV library such as CascadeClassifier and detectMultiScale for face detection by haarcascade_frontalface_default.xml.
  3. Unfortunately there was no way of using haarcascade_frontalface_default.xml without hardcoding the absolute path of the file on Mac. Initial accuracy resulted in close to 78%. With minor tweaks such as changing it to haarcascade_frontalface_alt.xml, changing minimum neighbors from 3 to 4, moving the scale factor between 1.1 to 1.5 and back to 1.1 where my best accuracy came about close to 81.5%.
  4. There were some obstacles that came in the way such as JSON unable to serialize int32 types of data in box so had to manually typecast the x,y,h,w values to int().
  5. With this setup, successfully created a model that can detect faces by drawing a bounding box around the face and extracting the coordinates pushed to json files such as results_val.json for validation images and results.json for test images.

- **Part B: Procedure & Challenges:**

1. Similar to the previous section, approached this with an intention to implement the K-means algorithm from scratch. Although it was successful in terms of implementation, the performance was unsatisfactory. Took 5 random centroids, from mean and standard deviation of encoded vector. Using the encoded vector, we calculate the distances from center points and form clusters. Iteratively we update the new center points by averaging all the points of the particular cluster and repeat the process for epoch number of times, here I tried 100.
2. The reasons for unsatisfactory performance was due to the fact that there was a 50% chance the cluster outputs were accurate while the other 50% gave bad results such as combining two clusters, producing less than 5 clusters, forming NaN valued arrays whenever distances were calculated therefore breaking the algorithm as a whole. Hence I took the low risk approach again to use scikit-learn K-means library which gave an accurate output.
3. To fix my initial implementation, wrote code blocks to handle errors thrown and repeat the K-means process whenever an undesired output was produced. One was an IndexError thrown whenever a NaN values in distances array and RuntimeError whenever there were less than 5 clusters produced. Eg:

```
while True:
        try:
            clusters = KMeans(int(K), encoded)
        except RuntimeWarning:
            continue
        break
```

4. In addition to this, whenever it produced less than 5 clusters, say 4, it was because one of the cluster was a combination of two actors that resulted in a size greater than 9. Since sizes of each cluster were between 5-9, wrote another code to rerun K-means whenever a cluster size exceeded 9. While this did fix my problem, this was only applicable to the bunch of images given to us and the algorithm could probably fail on a new dataset if it produces a cluster beyond 9 images and throw an unnecessary error. Hence the backup option was to take the simpler approach reputed to give an accurate result- Scikit-learn.
5. One of the biggest obstacles in this project was importing and using face-detection which happened to be a common problem among Mac M1 users. The error thrown was “Illegal Hardware Instruction” mentioned in Piazza post @734 which did temporarily fix my issue of creating a venv environment and reinstalling the libraries. With this, I was unable to import the sklearn library on Mac with the new environment due to a permissions issue so finished executing the project on a Windows laptop.
6. Hence, in the end, produced a cluster of images of 5 characters successfully using the K-means Algorithm.