# CV Project

## Attribute and Simile Classifiers for Face Verification


### **Team details**:

Name: sepnu
Members:
1. Loay Rashid (2018102008) - Team Leader
2. Tanmay Pathak (2018102023)
3. Haripraveen Subramanian (2018102031)
4. Pratikkumar Bulani (2019201074)

Teaching Assistant: Saraansh Tandon
Professor: Dr. Anoop Namboodiri


### **Overview of the project**:

1. Low Level Feature extraction: In this part, we extract a vector corresponding to each face region of a face.
2. Attribute Classifier: The extracted vectors for a face is checked for the presence of various attributes like Male, Attractive, Asian, Indian, etc. and to what extent.
3. Simile Classifier: The extracted vectors for a face is checked for the similarity of face regions with the reference people.
4. Verification Classifier: Two faces $F_1$ and $F_2$ are passed through all the learnt attribute classifiers and simile classifiers to obtain a trait vector for each face. Now verification classifier is trained to find whether both these trait vectors belong to the same person's face or not.

### **Datasets used**:

1. LFW: Used to train the attribute classifier and verification classifier.
2. CelebA: Used to train the attribute classifier.
3. Celebrity Face Recognition Dataset: Used to train the simile classifier.

### **Low Level Feature extraction details**:

1. Face Landmark Detection:
   a. We have used the pretrained model shape_predictor_81_face_landmarks.dat
   b. This model was trained on Helen dataset.
   c. To use this pretrained model, we have used the inbuilt dlib library
   d. The landmark positions are as shown:

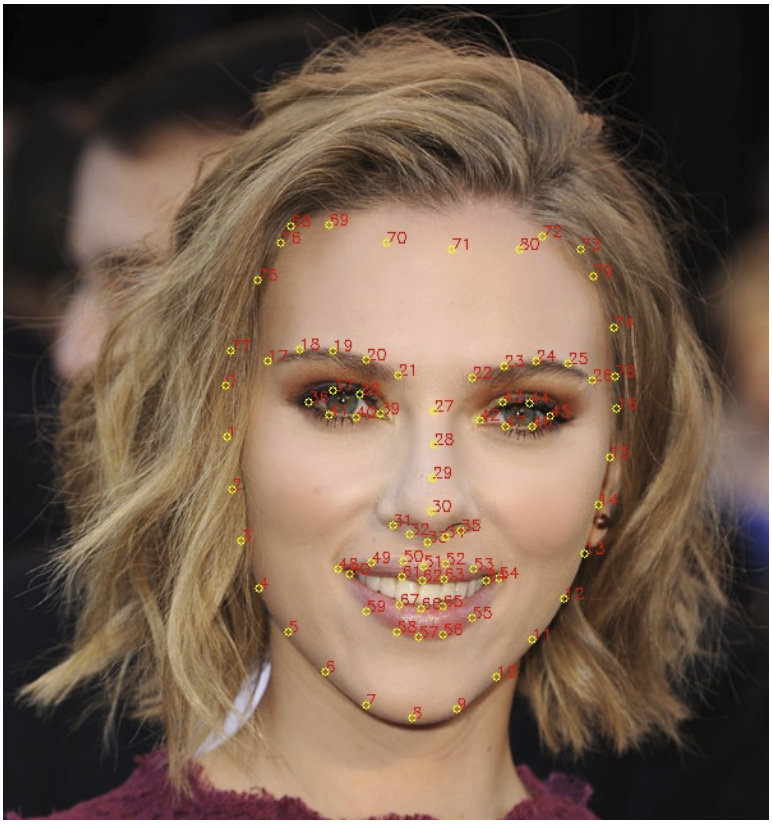


*Figure 1. Landmark locations and their indices*

2. Face alignment:
   a. Using the indices shown in figure 1, we have extracted the following regions: left eye and right eye.
   b. Using these regions, we can find the center of the left eye and center of the right eye.
   c. Taking the mean of the center of the left eye and the center of the right eye gives us the center of the forehead.
   d. The points: center of left eye, center of right eye and center of forehead, all are collinear i.e. lie on a line. We need to make this line parallel to the horizontal axis.
   e. To perform this parallelization, we perform affine warping of the entire image i.e. we rotate the image by an angle $\theta$ with the center of rotation equal to the center of forehead. Here $\theta$ = half of the slope of the line passing through the points: center of left eye, center of right eye and center of forehead. Examples of affine warping are shown:

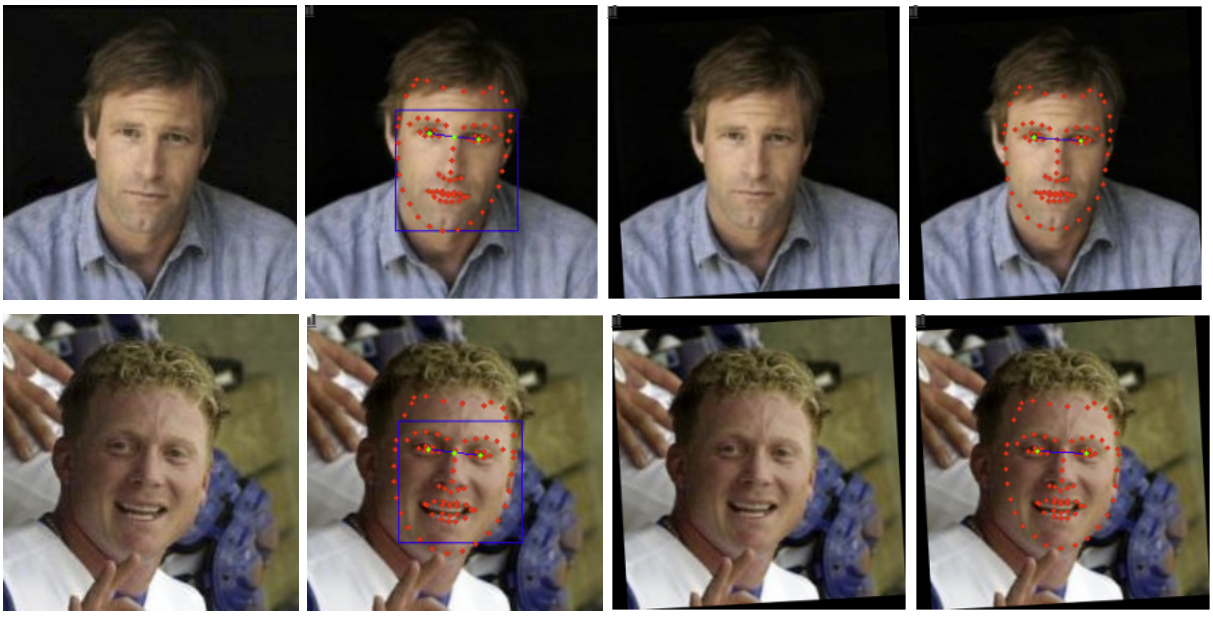


*Figure 2. Rotated Faces*

f. While extracting the face, we ensure that the center of the left eye falls at (0.35, 0.35) and center of the right eye falls at (0.65, 0.35). This assumes that we consider the image to be of size (1, 1). But the image size is (desired_height, desired_width). So, we scale the image while performing affine warp using the scale factor = (0.65 - 0.35) * desired_width / distance between the center of left and right eyes.

*Figure 3. Aligned Faces*

g. Thus, now all the faces are aligned to a common coordinate system. Since the face is affine warped, so should be the case with the landmark positions. Else there will be a mismatch between the affine warped face and landmark points.

3. Face region extraction:
   a. Using the indices shown in figure 1, we have extracted the following regions: left eye with eyebrow, right eye with eyebrow, nose, mouth, chin, moustache, left cheek, right cheek, forehead, hair and full face. Demonstration of extracted face regions:

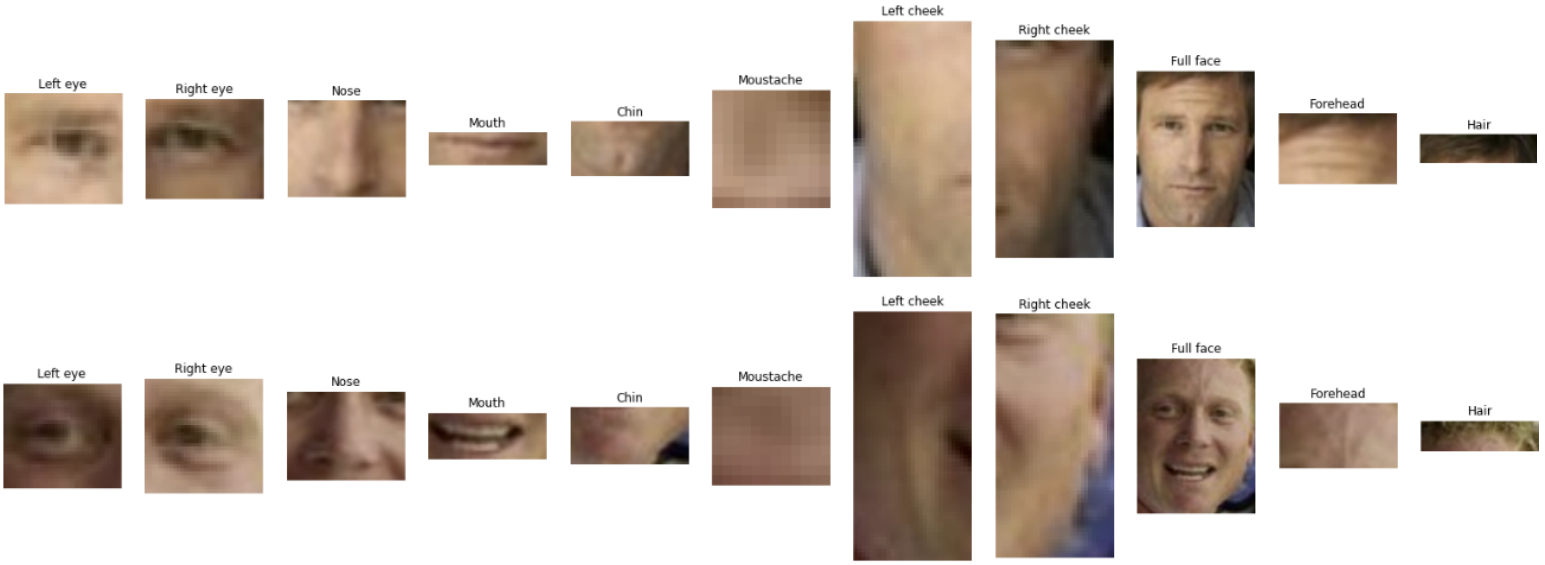



*Figure 4. Face regions*

4. Low level feature extraction:
   a. Each bounding box over a face region is expanded by some percentage. This is done so that the face pose is taken into consideration.
   b. For each face region, the image of that face region is normalized using standard preprocessing.
   c. Four different spaces are calculated for each face region: RGB color space, HSV color space, Edge magnitude space and Edge orientation space.
   d. The image is already in RGB space.
   e. We used the cv2 library to convert the image to HSV space
   f. Edge magnitude and Edge orientation space is calculated using cv2 inbuilt sobel filter. Edge magnitude is obtained using $\sqrt[2]{(sobel_x^2 + sobel_y^2)}$ and edge orientation is obtained using $\tan^{-1}\left(\frac{sobel_y}{sobel_x}\right)$.
   For saving space, we are just showing edge magnitude and edge orientation for full faces. But in code, we have calculated this for all the face regions.

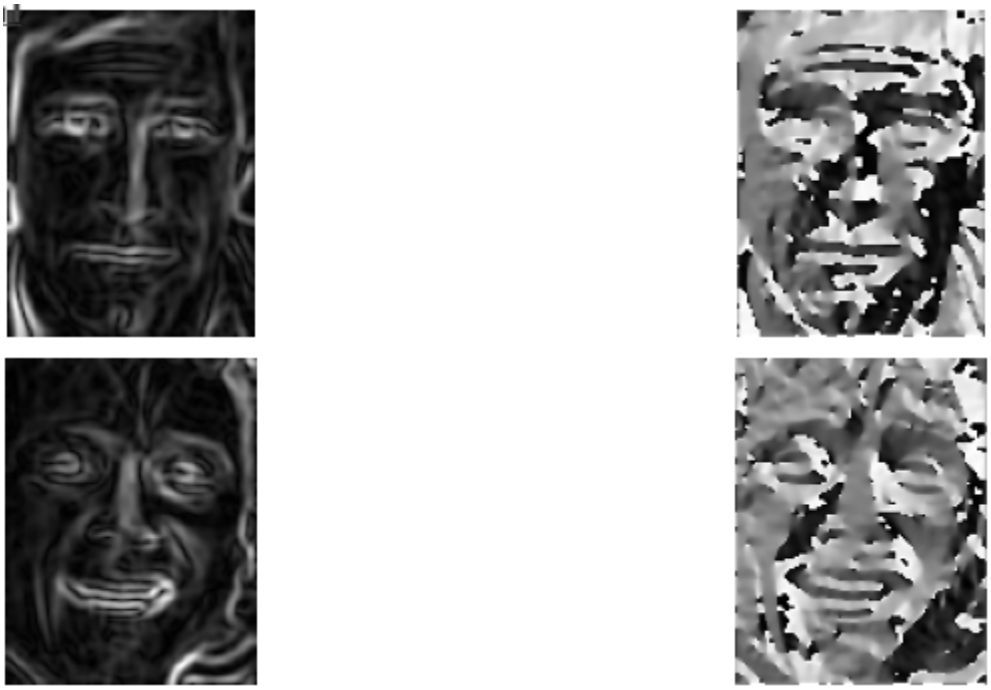


*Figure 5. Edge magnitude and Edge orientation space*

g. For each face region and each space taken into consideration, histogram is calculated over 100 bins.
h. These histograms are then normalized by dividing by the total number of points.
i. For each image, these calculated histograms are to be appended. But, we didn't append them, we saved them separately as .npy files.

## Attribute Classifier details:

We have used two separate datasets (CelebA and LFW) to check for accuracy improvements.
Because of lack of computation resources, we just used first 20,000 of the CelebA dataset. Due to lack of space on Google Drive, we used the pre-aligned CelebA images available officially.

Procedure:

1. We loaded the saved histograms.
2. These histograms are divided into train dataset and validation dataset using test size = 0.3
3. We have picked some handful of attributes from the given set of attributes. These attributes are chosen based on the face verification problem.
4. For each chosen attribute, we chose some features using which we have trained our SVM classifiers.
   Chosen attributes and chosen features for LFW dataset:

```
Male:grad_mag_full_face grad_orien_full_face
Asian:rgb_full_face hsv_full_face grad_mag_full_face grad_orien_full_face
White:rgb_forehead hsv_forehead
Black:rgb_forehead hsv_forehead
Bald:grad_mag_hair grad_orien_hair
Chubby:grad_mag_left_cheek grad_orien_left_cheek grad_mag_right_cheek grad_orien_right_cheek grad_mag_chin grad_orien_chin
Bushy Eyebrows:rgb_left_eye hsv_left_eye grad_mag_left_eye grad_orien_left_eye rgb_right_eye hsv_right_eye grad_mag_right_eye grad_orien_right_eye
Arched Eyebrows:grad_mag_left_eye grad_orien_left_eye grad_mag_right_eye grad_orien_right_eye
Big Nose:grad_mag_nose grad_orien_nose
Pointy Nose:grad_mag_nose grad_orien_nose
Round Jaw:grad_mag_chin grad_orien_chin grad_mag_left_cheek grad_orien_left_cheek grad_mag_right_cheek grad_orien_right_cheek
Oval Face:grad_mag_full_face grad_orien_full_face
Square Face:grad_mag_full_face grad_orien_full_face
Round Face:grad_mag_full_face grad_orien_full_face
Attractive Man:rgb_full_face hsv_full_face grad_mag_full_face grad_orien_full_face
Attractive Woman:rgb_full_face hsv_full_face grad_mag_full_face grad_orien_full_face
Indian:rgb_full_face hsv_full_face grad_mag_full_face grad_orien_full_face
Bags Under Eyes:rgb_left_eye hsv_left_eye grad_mag_left_eye grad_orien_left_eye rgb_right_eye hsv_right_eye grad_mag_right_eye grad_orien_right_eye
High Cheekbones:grad_mag_left_cheek grad_orien_left_cheek grad_mag_right_cheek grad_orien_right_cheek
Brown Eyes:rgb_left_eye hsv_left_eye rgb_right_eye hsv_right_eye
```

Chosen attributes and chosen features for CelebA dataset:

```
Arched_Eyebrows:grad_mag_left_eye grad_orien_left_eye grad_mag_right_eye grad_orien_right_eye
Attractive:rgb_full_face hsv_full_face grad_mag_full_face grad_orien_full_face
Bags_Under_Eyes:rgb_left_eye hsv_left_eye grad_mag_left_eye grad_orien_left_eye rgb_right_eye hsv_right_eye grad_mag_right_eye grad_orien_right_eye
Bald:rgb_hair hsv_hair grad_mag_hair grad_orien_hair
Black_Hair:rgb_hair hsv_hair
Blond_Hair:rgb_hair hsv_hair
Brown_Hair:rgb_hair hsv_hair
Bushy_Eyebrows:rgb_left_eye hsv_left_eye grad_mag_left_eye grad_orien_left_eye rgb_right_eye hsv_right_eye grad_mag_right_eye grad_orien_right_eye
Chubby:grad_mag_full_face grad_orien_full_face
Gray_Hair:rgb_hair hsv_hair
High_Cheekbones:grad_mag_full_face grad_orien_full_face
Male:grad_mag_full_face grad_orien_full_face
Oval_Face:grad_mag_full_face grad_orien_full_face
Pale_Skin:rgb_forehead hsv_forehead
Pointy_Nose:grad_mag_nose grad_orien_nose
Receding_Hairline:rgb_hair hsv_hair grad_mag_hair grad_orien_hair
Straight_Hair:grad_mag_hair grad_orien_hair
Wavy_Hair:grad_mag_hair grad_orien_hair
Young:rgb_full_face hsv_full_face grad_mag_full_face grad_orien_full_face
```

*Figure 6. Attributes and the chosen features*

5. While training SVMs, we have performed hyperparameter tuning using grid search.
6. After the model is trained, the models are saved.
7. The models are now tested for accuracy over the validation dataset.

Model Results: We have shown images of classification reports of some of our models which gave good accuracy. Later, we have given a table which gives the accuracy of all the trained models.
For LFW: All the classification reports for all the trained attribute classifiers are available in the .ipynb file.

```
----------For attribute Male----------
              precision    recall  f1-score   support

          -1       0.68      0.28      0.40       861
           1       0.83      0.96      0.89      3044

    accuracy                           0.81      3905
   macro avg       0.75      0.62      0.64      3905
weighted avg       0.79      0.81      0.78      3905
```

```
----------For attribute White----------
              precision    recall  f1-score   support

          -1       0.58      0.03      0.06       974
           1       0.75      0.99      0.86      2931

    accuracy                           0.75      3905
   macro avg       0.67      0.51      0.46      3905
weighted avg       0.71      0.75      0.66      3905
```

```
----------For attribute Bald----------
              precision    recall  f1-score   support

          -1       0.91      0.99      0.95      3505
           1       0.52      0.12      0.20       400

    accuracy                           0.90      3905
   macro avg       0.71      0.55      0.57      3905
weighted avg       0.87      0.90      0.87      3905
```

*Figure 7. Classification report for some of the attribute classifiers for LFW*

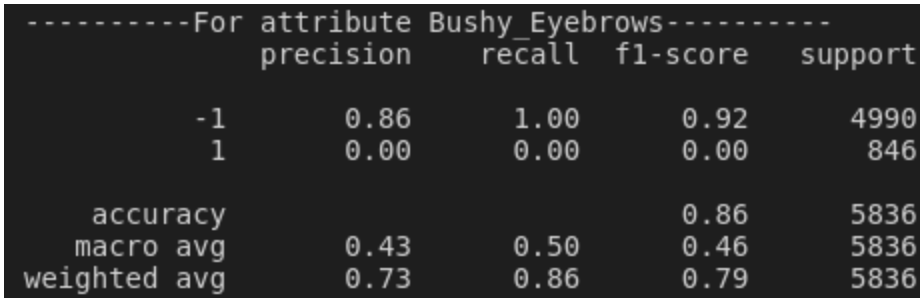
| Attribute | Accuracy Score |
|---|---|
| Male | 0.81 |

| | |
|---|---|
| Asian | 0.92 |
| White | 0.75 |
| Black | 0.96 |
| Bald | 0.90 |
| Chubby | 0.70 |
| Bushy Eyebrows | 0.66 |
| Arched Eyebrows | 0.77 |
| Big Nose | 0.73 |
| Pointy Nose | 0.73 |
| Round Jaw | 0.87 |
| Oval Face | 0.63 |
| Square Face | 0.95 |
| Round Face | 0.92 |
| Attractive Man | 0.68 |
| Attractive Woman | 0.85 |
| Indian | 0.98 |
| Bags Under Eyes | 0.71 |
| High Cheekbones | 0.75 |
| Brown Eyes | 0.67 |

For CelebA: All the classification reports for all the trained attribute classifiers are available in the .ipynb file.

```
----------For attribute Bald----------
              precision    recall  f1-score   support

          -1       0.98      1.00      0.99      5720
           1       0.41      0.16      0.23       116

    accuracy                           0.98      5836
   macro avg       0.70      0.58      0.61      5836
weighted avg       0.97      0.98      0.97      5836
```

```
----------For attribute Black_Hair----------
              precision    recall  f1-score   support

          -1       0.84      0.94      0.89      4424
           1       0.70      0.46      0.55      1412

    accuracy                           0.82      5836
   macro avg       0.77      0.70      0.72      5836
weighted avg       0.81      0.82      0.81      5836
```



```
----------For attribute Bushy_Eyebrows----------
              precision    recall  f1-score   support

          -1       0.86      1.00      0.92      4990
           1       0.00      0.00      0.00       846

    accuracy                           0.86      5836
   macro avg       0.43      0.50      0.46      5836
weighted avg       0.73      0.86      0.79      5836
```

*Figure 8. Classification report for some of the attribute classifiers for CelebA*

| Attribute | Accuracy Score |
|---|---|
| Arched_Eyebrows | 0.78 |
| Attractive | 0.67 |
| Bags_Under_Eyes | 0.79 |
| Bald | 0.98 |
| Black_Hair | 0.82 |
| Blond_Hair | 0.89 |
| Brown_Hair | 0.79 |
| Bushy_Eyebrows | 0.86 |
| Chubby | 0.94 |
| Gray_Hair | 0.96 |
| High_Cheekbones | 0.70 |
| Male | 0.76 |
| Oval_Face | 0.73 |
| Pale_Skin | 0.96 |
| Pointy_Nose | 0.72 |
| Receding_Hairline | 0.92 |
| Straight_Hair | 0.79 |
| Wavy_Hair | 0.69 |
| Young | 0.79 |

Why didn't we merge the dataset while training the attribute classifier? As there were less attributes common between both the dataset that we chose for face verification.

### **Simile Classifier details**:

We have used the dataset Celebrity Face Recognition. We have manually picked 37 celebrities and around 200 images for each celebrity. While manually selecting images, we forgot to incorporate the variations in face pose.

This made the simile classifier perform better on test data but we got inaccurate results while training the verification classifier.

Why did we manually pick the images? As there were a lot of erroneous images in the dataset. So we decided to manually clear those images.

Procedure:

1. We loaded the saved histograms
2. We generated labels for this dataset. If the histograms belong to the same person, then the label is +1. Else it is -1.
3. The dataset is divided into train data and test data. The train labels -1 is divided using the test size = 0.3 and the train labels +1 is divided using the test size = 0.15
4. We have chosen some handful of reference persons from the dataset.

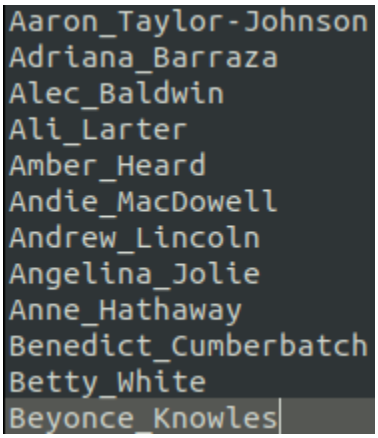



*Figure 9. Chosen reference persons*

5. We have trained SVM models for the face regions: eyes, nose and mouth. We have chosen the following features to train:

```
{"eyes":["rgb_left_eye", "hsv_left_eye", "grad_mag_left_eye", "grad_orien_left_eye", "rgb_right_eye", "hsv_right_eye", "grad_mag_right_eye", "grad_orien_right_eye"],
 "nose":["rgb_nose", "hsv_nose", "grad_mag_nose", "grad_orien_nose"],
 "mouth":["rgb_mouth", "hsv_mouth", "grad_mag_mouth", "grad_orien_mouth"]}
```

*Figure 10. Regions and their features chosen for Simile Classifier*

6. While training SVMs, we have performed hyperparameter tuning using grid search.
7. After the model is trained, the models are saved.
8. The models are now tested for accuracy over the validation dataset.

Results: Since there is no face pose variation in the manually picked dataset, we got very high accuracy in almost most of the cases.

```
----------For reference person Aaron_Taylor-Johnson and attribute eyes----------
              precision    recall  f1-score   support

          -1       0.99      1.00      0.99      2124
           1       0.53      0.29      0.38        31

    accuracy                           0.99      2155
   macro avg       0.76      0.64      0.68      2155
weighted avg       0.98      0.99      0.98      2155

----------For reference person Aaron_Taylor-Johnson and attribute nose----------
              precision    recall  f1-score   support

          -1       0.99      1.00      0.99      2124
           1       0.69      0.29      0.41        31

    accuracy                           0.99      2155
   macro avg       0.84      0.64      0.70      2155
weighted avg       0.99      0.99      0.99      2155

----------For reference person Aaron_Taylor-Johnson and attribute mouth----------
              precision    recall  f1-score   support

          -1       0.99      0.99      0.99      2124
           1       0.48      0.42      0.45        31

    accuracy                           0.99      2155
   macro avg       0.74      0.71      0.72      2155
weighted avg       0.98      0.99      0.98      2155
```

## Verification Classifier:

In this we took around 6100 examples 3000 negative and 3104 positive pairs of images from the lfw dataset and loaded the pretrained svms and took the output on the positive and negative examples.

Procedure:

1. We divided the dataset randomly into positive and negative images.
2. We extracted the low level features for these pairs of images.
3. We fetched the output of simile and attribute classifiers for these pairs of images (using the low level features extracted above).
4. The output of simile and attribute classifier from each pair of images is concatenated.
5. We generated labels for this dataset. If the outputs of simile and attribute classifiers belong to the same person, then the label is +1. Else it is 0.
6. We trained the SVM.

Results:

Using method 1:

```
              precision    recall  f1-score   support

          -1       0.71      0.69      0.72       853
           1       0.59      0.62      0.61       990

    accuracy                           0.66      1843
   macro avg       0.63      0.62      0.63      1843
weighted avg       0.65      0.66      0.65      1843
```

Using method 2:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| -1 | 0.72 | 0.69 | 0.71 | 887 |
| 1 | 0.61 | 0.63 | 0.62 | 956 |
| accuracy | | | 0.67 | 1843 |
| macro avg | 0.63 | 0.61 | 0.62 | 1843 |
| weighted avg | 0.66 | 0.67 | 0.64 | 1843 |

**Final Result**:

Some true positives of our classifier:



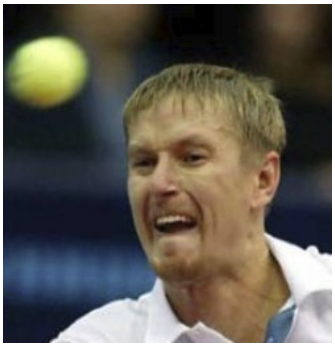



Some true negatives of our classifier:






Some false negatives of our classifier:

Some false positives of our classifier:






********** THE END **********
********** WAIT!! WE HAVE ALSO DONE THE BONUS **********

## BONUS

Since this paper is from 2009, it focused only upon classification using SVMs. We completed its implementations, and then tried to think of better techniques to improve verification accuracy.

So, we applied various deep learning techniques and in turn, gained a much better understanding of deep learning and how convolutional neural networks help in improving accuracy.

**Attribute Classifiers from Low-Level Histograms:**
We take the histograms generated using our low-level code (explained earlier) and train a basic linear neural network for multiple attributes (one NN for each attribute). This is equivalent to replacing the SVMs in the paper with neural nets. The results follow:

For Asian attribute:

```
Epoch :15
----------
train:
loss: 0.1867   acc:0.93847
Accuracy on the test images: 0.92852
```

For Male attribute:

```
Epoch :15
----------
train:
loss: 0.2531   acc:0.8390
Accuracy on the test images: 0.8247
```

This is a major improvement over SVMs in terms of training speed, and the accuracy is slightly better, but not significant. Also, this requires a large number of NNs to be saved, which takes up a huge amount of space. End-to-end result generation will also be slower.

### Convolutional Neural Networks:

Next, we tried to use convolutional networks to get better results.
We used an architecture loosely based on AlexNet and trained it on single attributes on the LFW dataset. On the 13014 images we achieved significant improvements in accuracy over plain SVMs.

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=12544, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=2048, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=2048, out_features=2, bias=True)
  )
)
```

For the male attribute:

```
==========================================================
Current Attribute: Male
train:
loss = 0.2058     accuracy = 0.9247
test:
loss: 0.2690     accuracy: 0.9132
```

We achieved much better than svms (79%)

For Asian attribute:

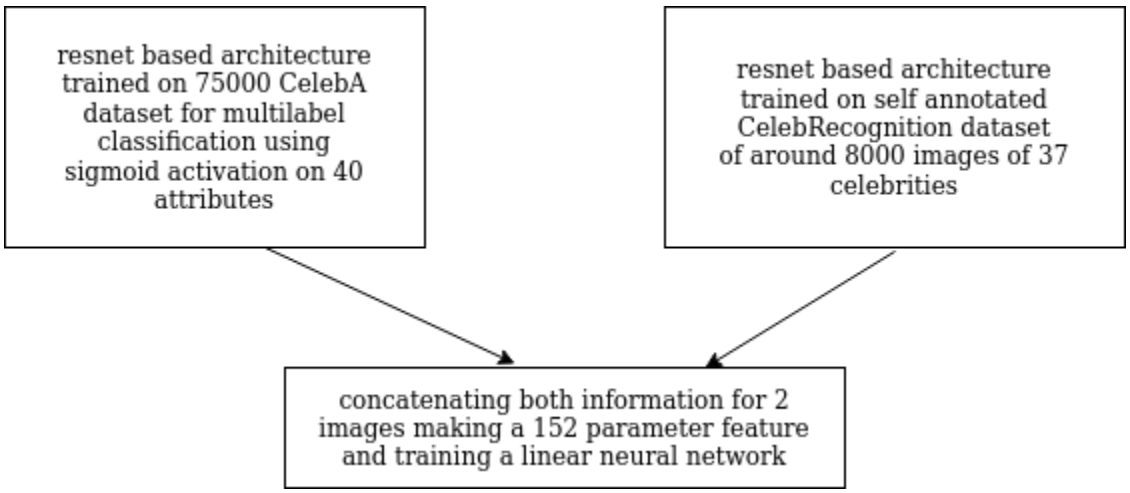
```
==========================================================
Current Attribute: Asian
train:
loss = 0.2290     accuracy = 0.9188
test:
loss: 0.2512     accuracy: 0.9063
```

Here we matched the svms(89%)

## Replicating the paper using deep convolutional networks:

We felt that accuracy could be improved even further using CNNs. So, we set out to replicate the paper using CNNs.

Consider the following diagram:




## Multilabel Attribute Classifier Neural Network

We randomly selected 75000 images of the CelebA and made an extended resnet based architecture that we modified for a multilabel classification. Since one person can have multiple attributes, we decided to drop the log

softmax activation function and instead opted for sigmoid activation. Moreover, we changed the loss function from Cross Entropy Loss to Binary Cross Entropy Loss.

We evaluated our model based on how many correct labels our model predicted and summing the number of correct label predictions for each image and then dividing it by the number of labels.

**Procedure:**

1. We created a custom dataset for CelebA and reduced the number of images from a total of 202k.
2. Then we took a backbone of the resnet50 neural network and defined new layers for 40 attributes. Since we had to do multilabel classification we tried a few different methods. First, we made multiple heads for each attribute with CE loss. This was hard to work with, not scalable, and didn't give us satisfactory results. So, we removed the heads and tried to generate the outputs of all 40 attributes at once using a sigmoid activation at the end, and changed the loss to Binary cross entropy.
3. At the output, we round the output to 0 or 1 and then check with the input labels and classify which are correct.
4. Finally we ran this using the ADAM optimizer for 40 epochs at a learning rate of 1e-3.
5. For 40 attributes we achieved an accuracy of 82%

**Improvements:**

- We generate the outputs of all 40 attributes at once (through one model). This takes up significantly lesser space and gives us much better speed.
- The Accuracy (82%) is better than that for SVMs and is comparable to that for individual attribute NNs.s

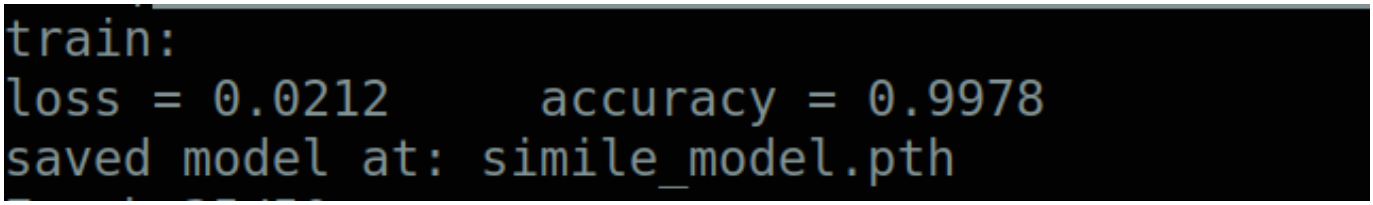
## Simile classifier:

We made a basic classification neural network with 37 outputs (reference people). The output is taken to be a one-hot vector because one person can only look like one celebrity. A pre-trained resnet was used because of the low number of training images (8100) We used the self-annotated CelebrityFaceRecognition dataset of 37 celebrities.

Procedure:

1. We created a custom dataset and labeled the outputs from 0 to 36.
2. We then took a pre-trained resnet50 neural network and defined a new layer of 36 attributes. Here we had to do normal classification
3. At the outputs we take the max of all the vectors to 0 or 1 (since a person can look like only one celebrity) and then check with the input labels and classify which are correct.
4. Finally we trained the network using the ADAM optimizer for 50 epochs at a learning rate of 1e-3.
5. For 37 reference people we achieved an accuracy of 99%.

At the end of 50 epochs:




At the 32nd epoch:

```
Epoch 32/50
----------
100%|
train:
loss = 0.0887     accuracy = 0.9765
saved model at: simile_model.pth
```

**Improvements;**

- This is a MAJOR improvement over the simile SVMs in terms of accuracy. Moreover, this has the benefit of checking for all 37 reference people at a single time instead of repeatedly having to load an SVM for each person.

## Final verification neural network

Finally, we take the 36 length output from the Simile network and the 40 length output from the Attribute network. These are then concatenated to a single 76 length output, and since we consider two images for verification, 152 length training vectors were formed with 0,1 labels (1 meaning same and 0 means different). We built a relatively small neural network for this and trained it for 50 epochs. This gave us an accuracy of 78%. Major improvements are possible in this area.

```
VerifNet(
  (classifier): Sequential(
    (0): Linear(in_features=152, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=4096, out_features=2048, bias=True)
    (3): ReLU(inplace=True)
    (4): Dropout(p=0.5, inplace=False)
    (5): Linear(in_features=2048, out_features=1024, bias=True)
    (6): ReLU(inplace=True)
    (7): Dropout(p=0.5, inplace=False)
    (8): Linear(in_features=1024, out_features=512, bias=True)
    (9): ReLU(inplace=True)
    (10): Linear(in_features=512, out_features=2, bias=True)
  )
)
```

Limitations and Possible Improvements:

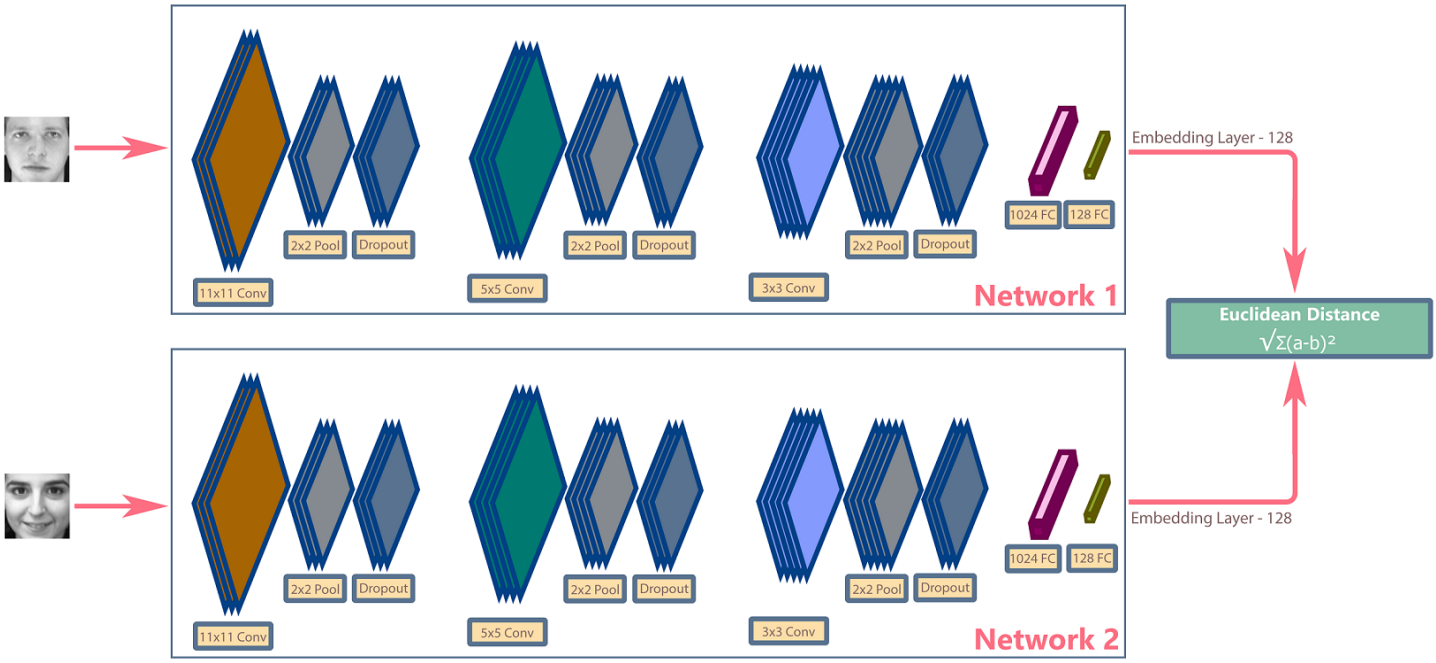
1. Lack of time and computation resources: We only had access to colab for the majority of the project and colab often kills due to inactivity and is not very powerful.
2. Fault in Dataset Selection: We later realized that the Celebrity Face Recog dataset had very good resolution and most of the images were perfect due to human selection (we picked images that weren't at an angle and had fully visible features). On the other hand, CelebA and LFW had very low res pictures and high variability in the data. This is why combining the results of SVM's (or neural networks) trained on such different datasets couldn't perform well.
3. Data augmentation: We could have added normalization and many transforms that pytorch offers in order to make our models more robust and further increase verification accuracy.

********** THE END **********

******* STILL WAIT!! WE HAVE ALSO DONE LITERATURE SURVEY ON SIAMESE NETWORK *******

**Face Verification using Siamese networks:**

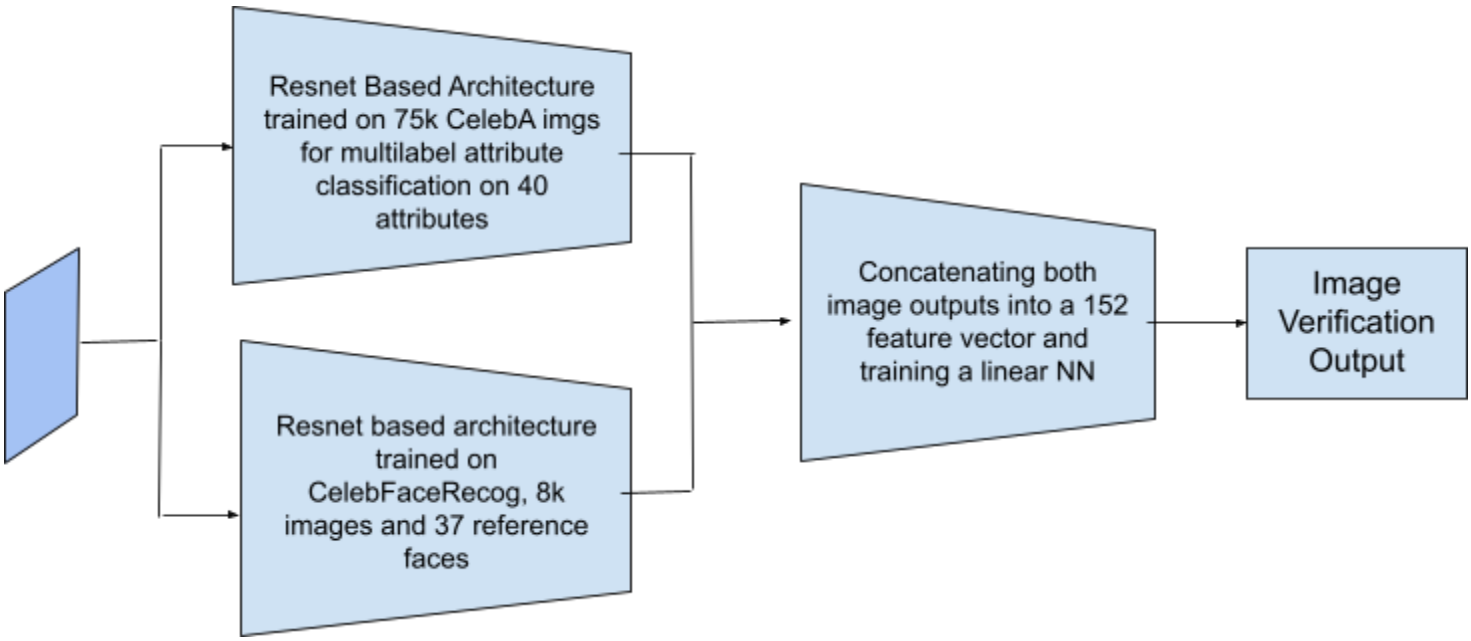
The Siamese network consists of a pair of Neural Networks which are identical to each other, also known as Sister Networks. The Siamese network is used to find the distance between any two given images. If the images have the same label, then the network should learn the parameters, i.e. the weights and the biases in such a way that it should produce a smaller distance between the two images, and if they belong to different labels, then the distance should be larger.




1. To train a Siamese Network, a pair of images are picked from the dataset, each one processed by one of the networks above.
2. Both the low-level feature extraction networks have the same architecture structure, hence the same operations will be performed on the respective images.
3. The Neural Networks at the end have Fully Connected Layers, with the last one consisting of 128 nodes. This layer is the final feature that gets produced when the network is applied on the image. It's called the Embedding Layer Representation. So the two images in the pair processed by the Siamese Network produce two different Embedding Layer Representations.
4. The Network then finds the Euclidean distance between both the embedding layers. If the images are of the same person, then it is expected that the embeddings will be very similar, hence distance should be smaller. However, if the images are of different people, then the distance is expected to be a higher value.
5. A Sigmoid Function is applied on the distance value to bring the value in the range [0, 1]
6. We can use the Binary Cross Entropy loss to update the weights and biases of the network. Updation of the weights and the biases done on both the networks are exactly the same.

********** THE END **********

********** THANKS FOR FOLLOWING ALONG WITH US SO FAR! BYE! **********


Resnet Based Architecture trained on 75k CelebA imgs for multilabel attribute classification on 40 attributes
Resnet based architecture trained on CelebFaceRecog, 8k images and 37 reference faces
Concatenating both image outputs into a 152 feature vector and training a linear NN
Image Verification Output