ATTRIBUTE AND SIMILE CLASSIFIERS FOR FACE VERIFICATION

Team name - Sepnu

Teaching Assistant: Saraansh Tandon

1) Loay Rashid (2018102008)
2) Tanmay Pathak (2018102023)
3) Haripraveen Subramanian (2018102031)
4) Pratikkumar Bulani (2019201074)

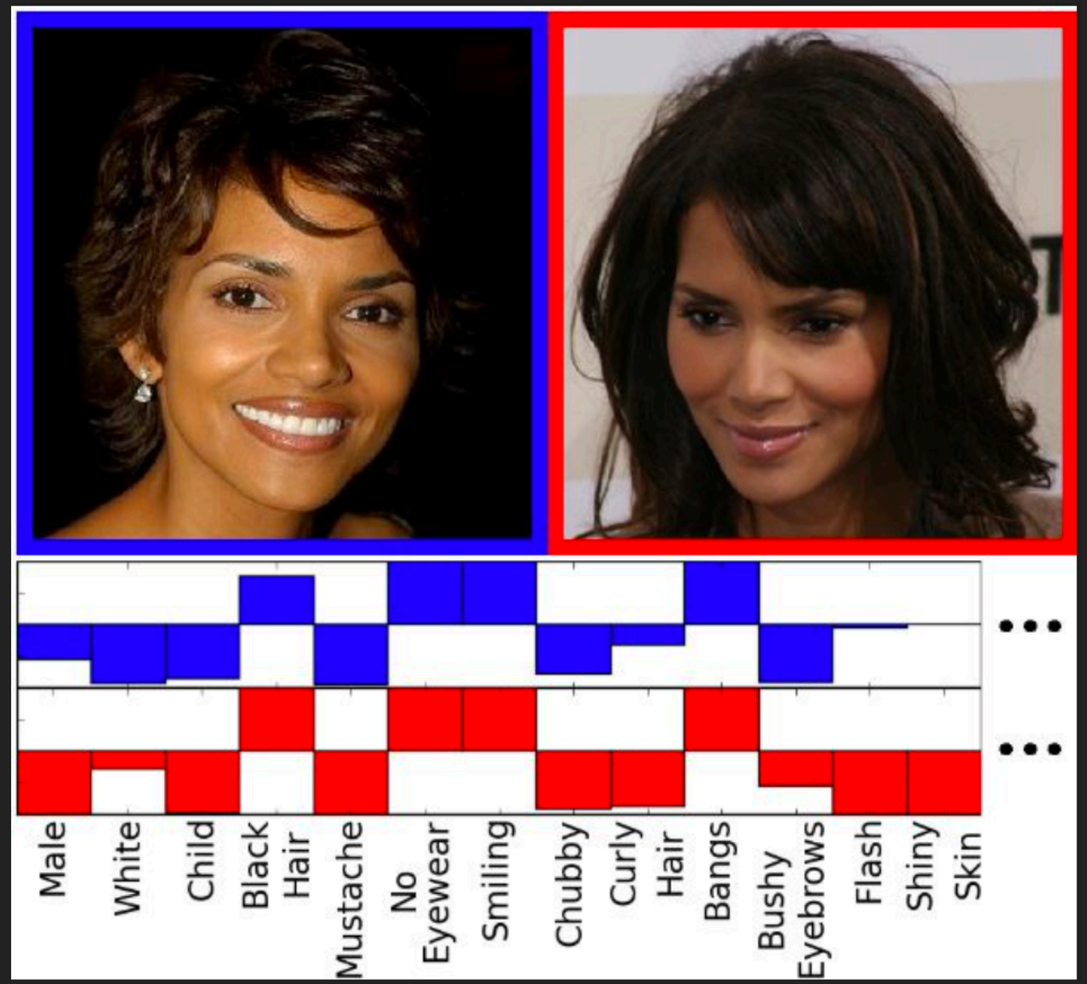
FACE RECOGNITION V/S VERIFICATION

- **Face recognition**: Scanning a face and matching it with a database of known faces.
- **Face verification:** Confirms that the physical face matches the one present on the ID document.
- This research paper presents two novel methods of face verification.

ATTRIBUTE CLASSIFIER

- Uses binary classifiers to recognize the presence or absence of descriptable aspects of visual appearance (gender, race, age, hair colour, etc.)
- The first histogram gives values of attributes for left image.
- The second histogram gives values of attributes for right image.

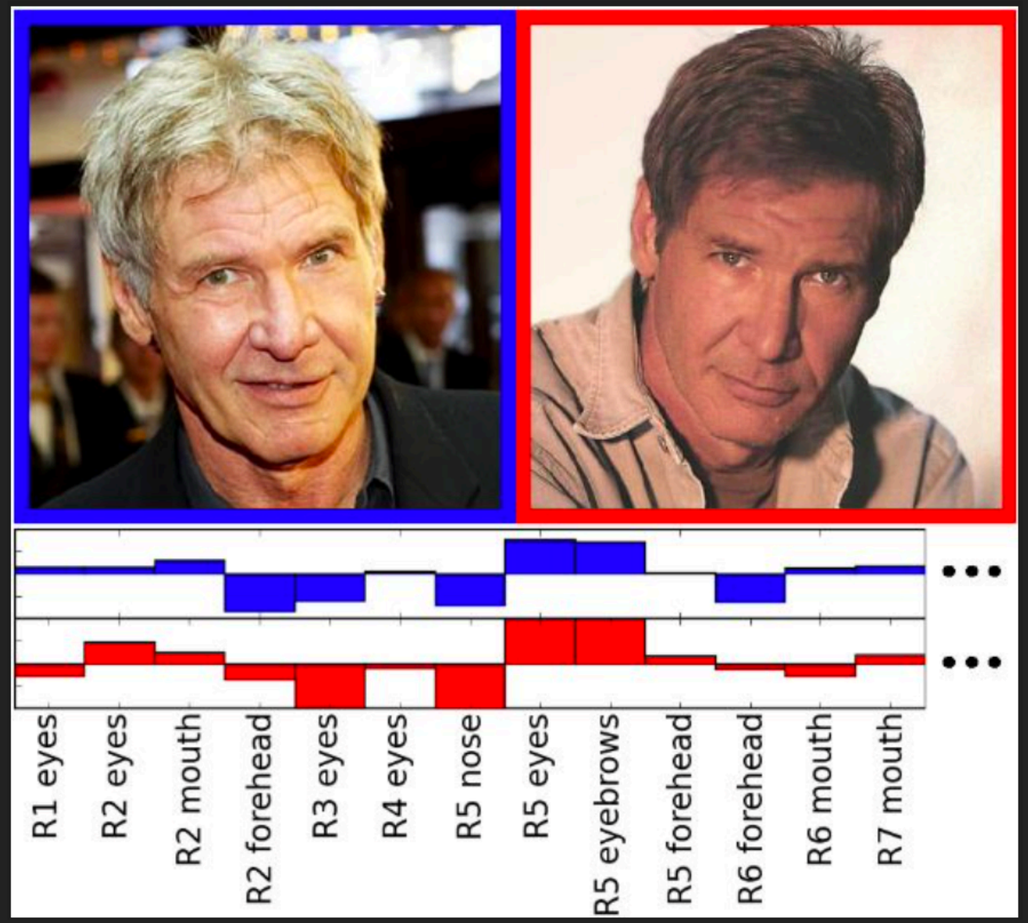
Examples of ATTRIBUTES ——————————————————>



SIMILE CLASSIFIER

- Removes the manual labelling required to train attribute classifiers. They are binary classifiers trained to recognize the similarity of faces, or regions of faces, to specific reference people.
- An unseen face might be described as having a mouth that looks like Barack Obama's and a nose that looks like Owen Wilson’s.

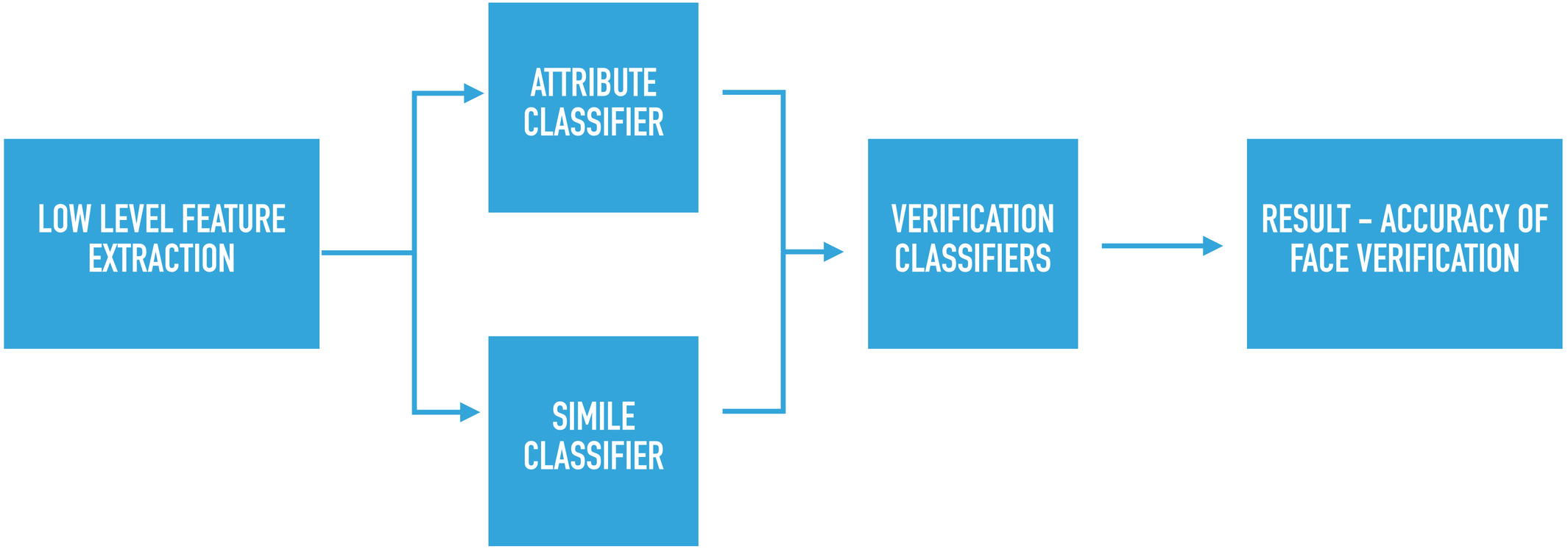
Examples of SIMILES ––––––––––––––––––––––––>



DATASETS USED

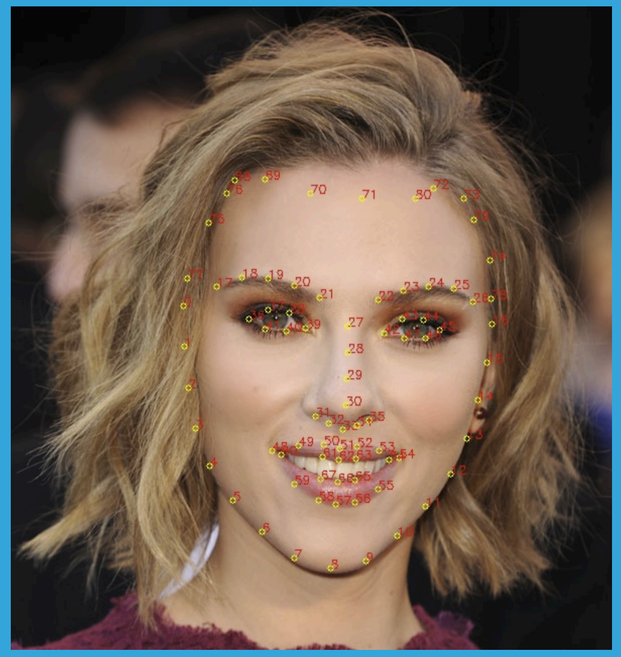
1. **LFW:** Used to train the attribute classifier and verification classifier.
    1. http://vis-www.cs.umass.edu/lfw/
2. **CelebA:** Used to train the attribute classifier.
    1. http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
3. **Celebrity Face Recognition Dataset:** Used to train the simile classifier.
    1. https://github.com/prateekmehta59/Celebrity-Face-Recognition-Dataset

OVERVIEW OF THE PROJECT



STEPS INVOLVED IN LOW LEVEL FEATURE EXTRACTION

STEPS INVOLVED IN LOW LEVEL FEATURE EXTRACTION

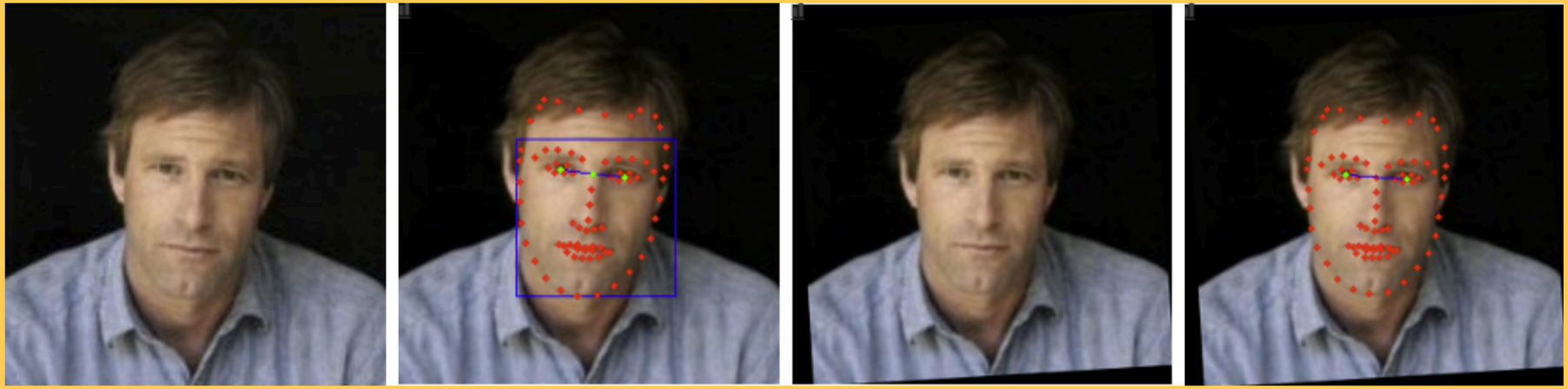


FACE LANDMARK DETECTION

1. We have used the pre-trained model shape_predictor_81_face_landmarks.dat
2. This model was trained on Helen dataset.
3. To use this pre-trained model, we have used the inbuilt dlib library
4. The landmark positions are as shown

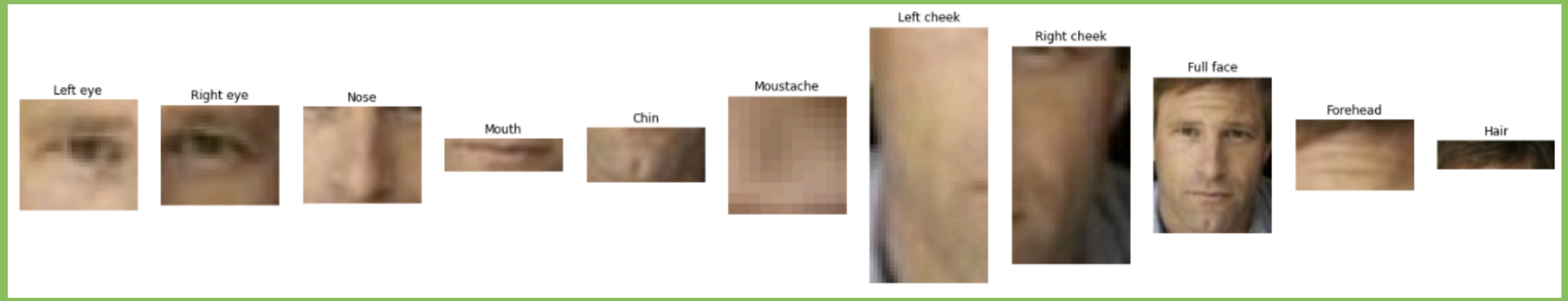
STEPS INVOLVED IN LOW LEVEL FEATURE EXTRACTION

FACE ALIGNMENT



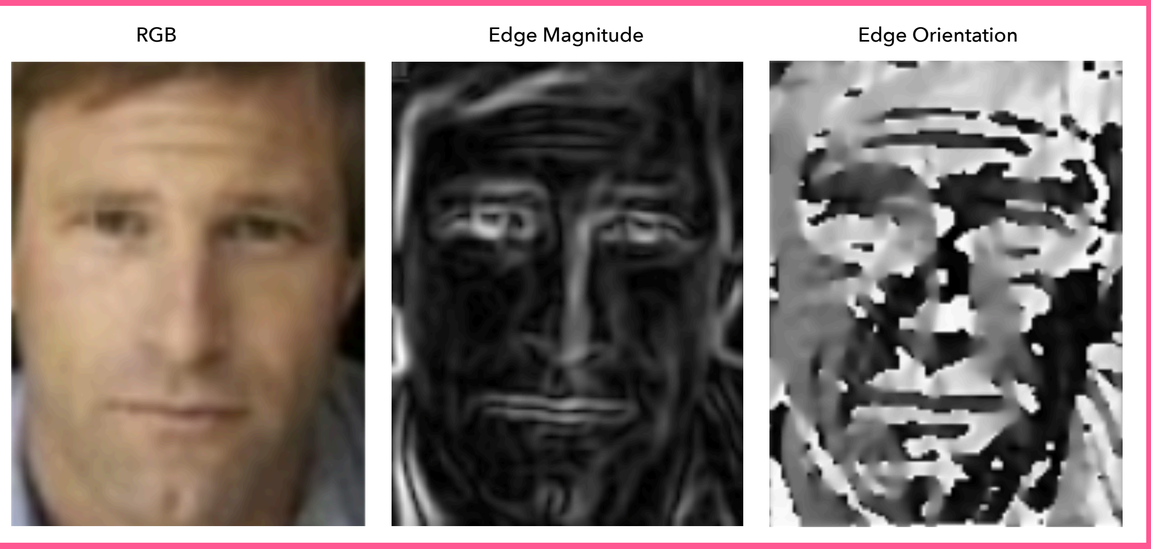
STEPS INVOLVED IN LOW LEVEL FEATURE EXTRACTION

FACE REGION EXTRACTION



STEPS INVOLVED IN LOW LEVEL FEATURE EXTRACTION

LOW LEVEL FEATURE EXTRACTION



ATTRIBUTE CLASSIFIER – STEPS INVOLVED

1. The saved low-level histograms were loaded.
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.
5. While training SVMs, we have performed hyper-parameter 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.

ATTRIBUTE CLASSIFIER – MODEL RESULTS (LFW DATASET)

```
----------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
```

Attribute	Accuracy Score
Male	0.81
Asian	0.92
White	0.75
Black	0.96
Bald	0.9
Chubby	0.7
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

ATTRIBUTE CLASSIFIER – MODEL RESULTS (CELEBA DATASET)

```
----------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
```

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	0.7
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

SIMILE CLASSIFIER – STEPS INVOLVED

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.
5. We have trained SVM models for the face regions: eyes, nose and mouth.
6. While training SVMs, we have performed hyper-parameter 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.

SIMILE CLASSIFIER – MODEL RESULTS (CUSTOM DATASET)

```
----------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
```

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

VERIFICATION CLASSIFIER – STEPS INVOLVED

1. Extracted 6100 examples (3000 negative and 3104 positive) pairs of images from the LFW dataset and loaded the pre-trained SVMs and took the output on the positive and negative examples
2. We divided the dataset randomly into positive and negative images.
3. We extracted the low level features for these pairs of images.
4. We fetched the output of simile and attribute classifiers for these pairs of images (using the low level features extracted above).
5. The output of simile and attribute classifier from each pair of images is concatenated.
6. 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.
7. We trained the SVM.

VERIFICATION CLASSIFIER – RESULTS

	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

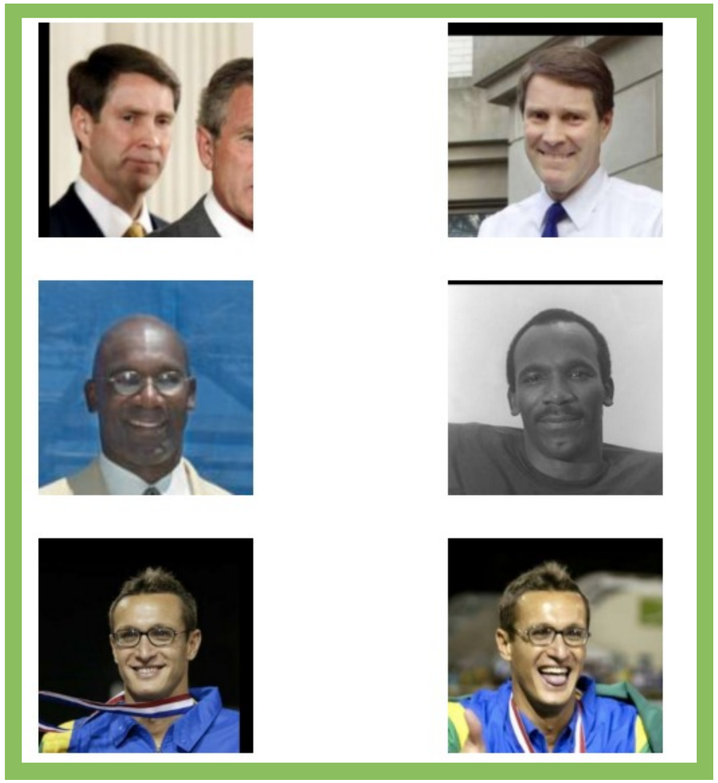
Method 1

	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

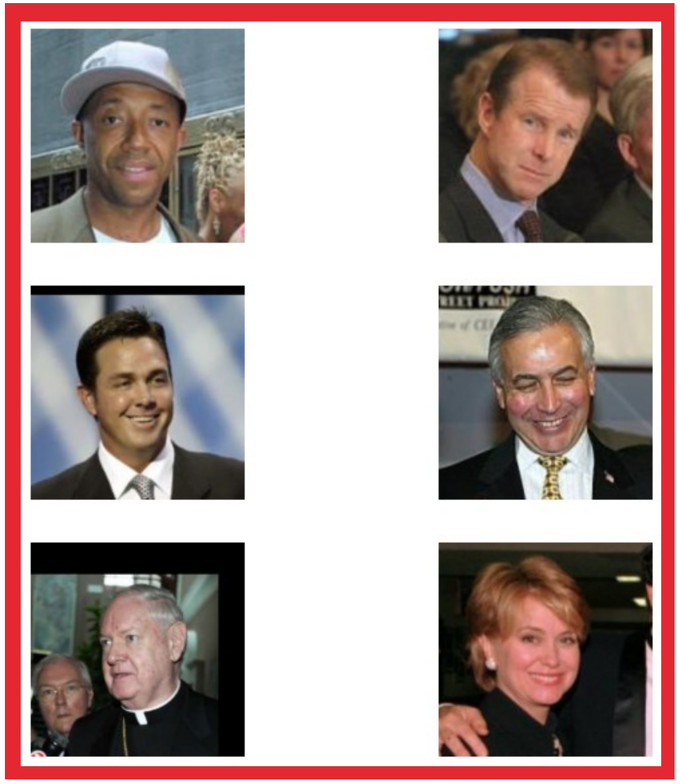
Method 2

Method 1: We did what the paper suggested and weighted the element-wise subtraction and multiplication of the trait vectors with a 0 mean, 1 variance Gaussian.

Method 2: We directly concatenated the element-wise trait vector subtraction and multiplication without any weighting. We get an extremely minute improvement in accuracy.



True Positives



True Negatives

VERIFICATION CLASSIFIER – RESULTS

False Positives

False Negatives

ONE MORE THING. . .

BONUS – INCORPORATING DEEP LEARNING

- 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.
- RESULT: We see that this is an improvement on the SVMs on the basis of training speeds and slightly on accuracy but still results weren't significant.

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

Asian Attribute

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

Male Attribute

CONVOLUTIONAL NEURAL NETWORK

- Now we tried to use convolutional networks to get better accuracy.
- We used an architecture loosely based on AlexNet and trained it on single attributes on the LFW dataset. On the 13014 images . We achieved much better accuracy than the 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)
  )
)
```

CONVOLUTIONAL NEURAL NETWORK – RESULTS

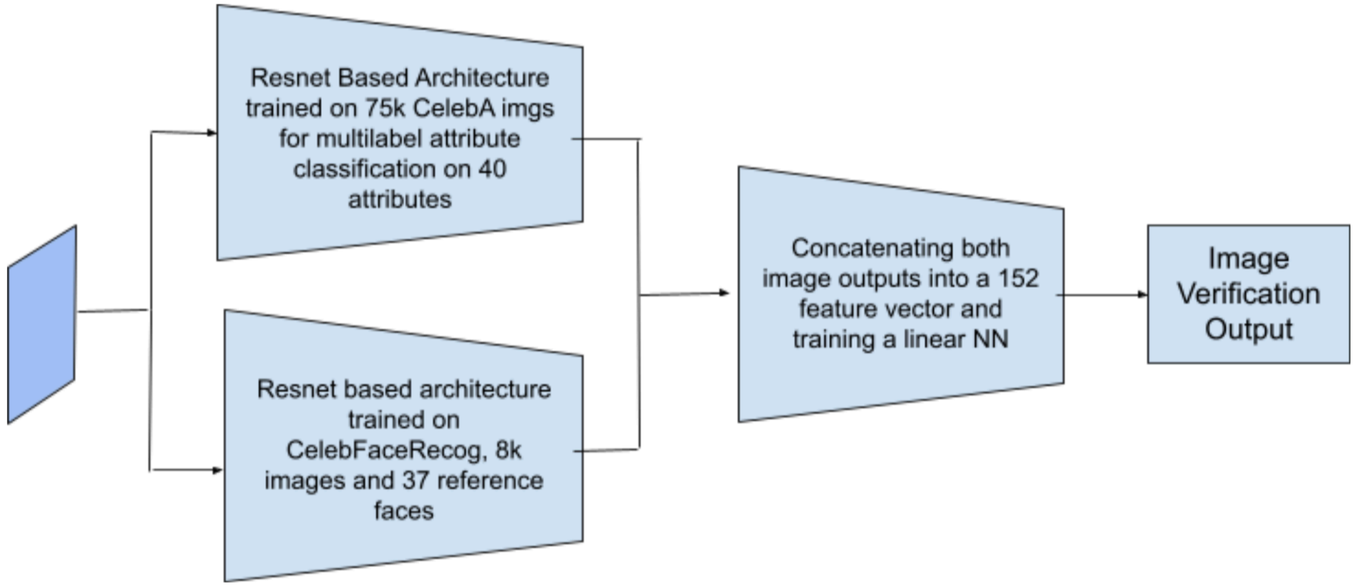
```
==========================================================
Current Attribute: Male
train:
loss = 0.2058     accuracy = 0.9247
test:
loss: 0.2690     accuracy: 0.9132
```

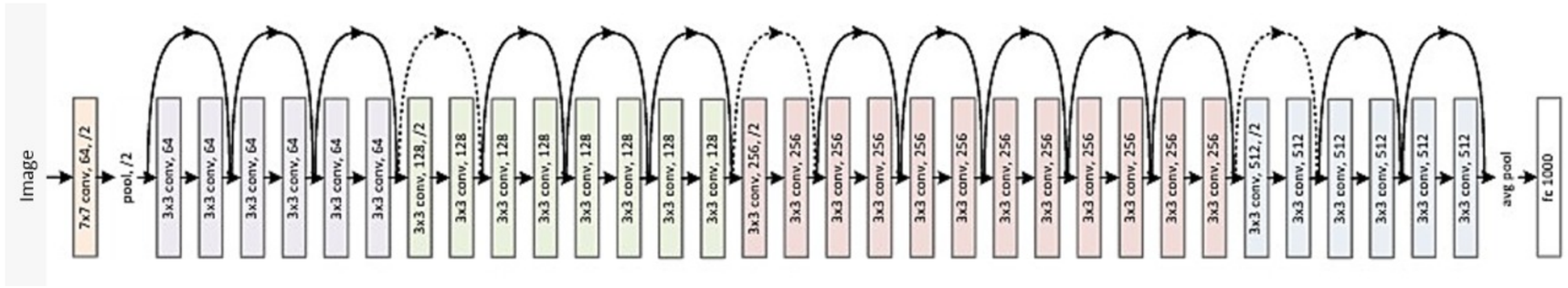
We achieved much better than SVMs (79%)

```
==========================================================
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





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 multi-label 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.

MULTILABEL ATTRIBUTE CLASSIFIER NEURAL NETWORK – STEPS INVOLVED

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 multi-label 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%

MULTILABEL ATTRIBUTE CLASSIFIER NEURAL NETWORK – RESULTS

- We notice some improvements over the results mentioned in the paper.
- 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.

SIMILE CLASSIFICATION USING NEURAL NETWORKS

- 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.

SIMILE CLASSIFICATION USING NEURAL NETWORKS – STEPS INVOLVED

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%.

SIMILE CLASSIFICATION USING NEURAL NETWORKS – RESULTS

- We again see 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.

```
train:
loss = 0.0887      accuracy = 0.9765
saved model at: simile_model.pth
```

At the end of 32 epochs

```
train:
loss = 0.0212      accuracy = 0.9978
saved model at: simile_model.pth
```

At the end of 50 epochs

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)
  )
)
```

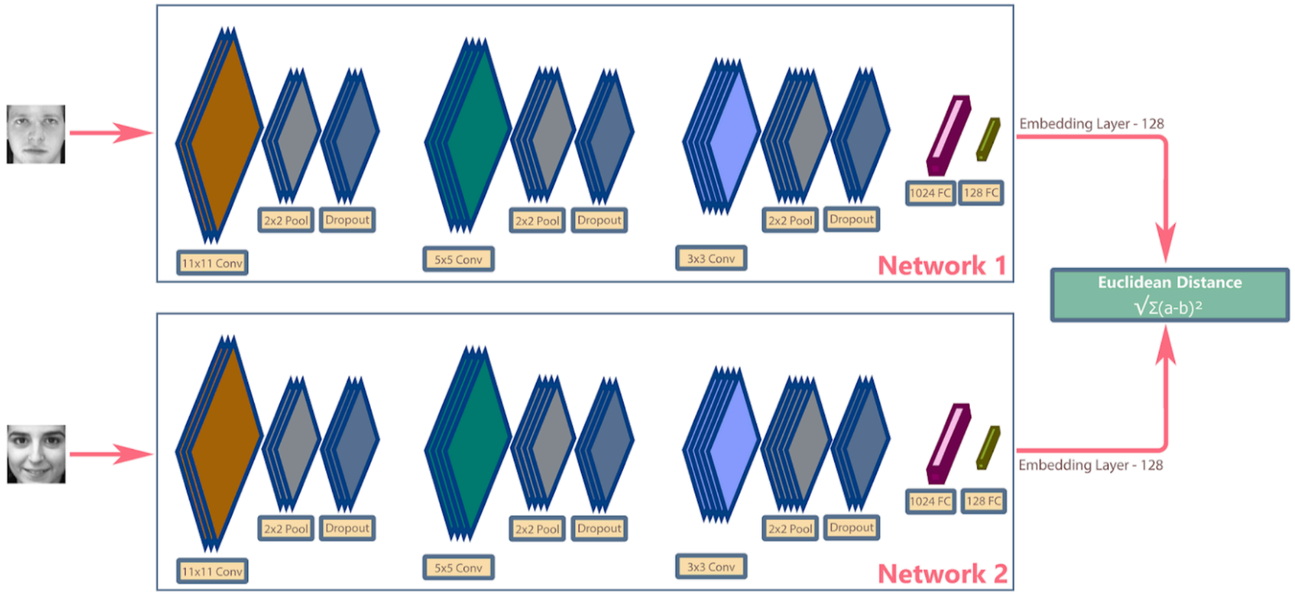
LIMITATIONS AND POSSIBLE IMPROVEMENTS

- **Lack of time and computation resources**: We only had access to google colab for the majority of the project and colab often kills due to inactivity and is not very powerful.
- **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.
- **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.

ADDITIONAL LITERATURE REVIEW

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.



SIAMESE NETWORKS – PROCEDURE

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.

THANK YOU