Attribute and Simile Classifiers for Face Verification

Team: Computer Visionaries

Ishaan Khare 20171153

Saraansh Tandon 20171007

Shubhankar Bhagwat 20171147

TABLE OF CONTENTS

INTRODUCTION

We present two novel methods for face verification. Our first method – “attribute” classifiers – uses binary classifiers trained to recognize the presence or absence of describable aspects of visual appearance (e.g., gender, race, and age). Our second method – “simile” classifiers – removes the manual labeling required for attribute classification and instead learns the similarity of faces, or regions of faces, to specific reference people.

1. **Attribute classifiers** - An attribute classifier can be trained to recognize the presence or absence of a describable aspect of visual appearance. We introduce classifiers for face verification, using 65 describable visual traits such as gender, age, race, hair color, etc.

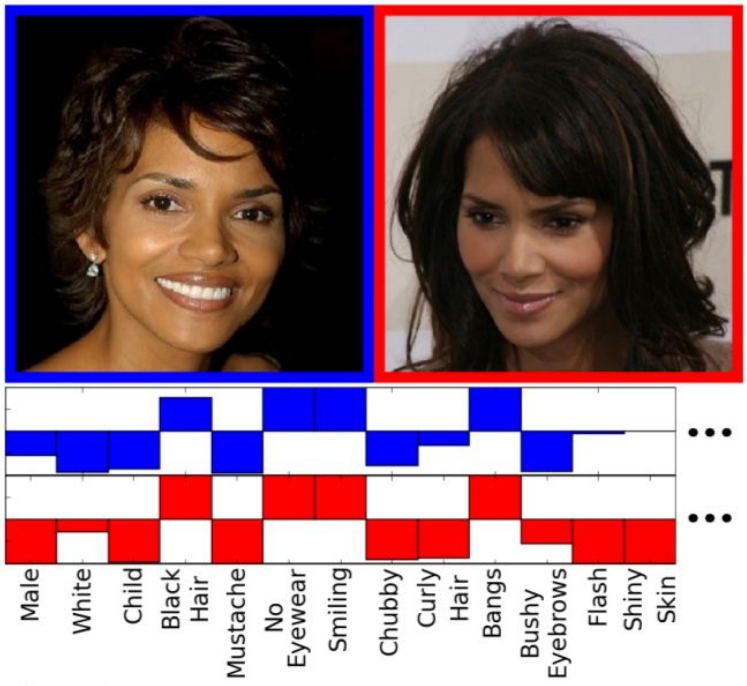


Figure 1 : Attribute classifier

2. **Simile classifiers** - We use a large number of “simile” classifiers trained to recognize the similarities of parts of faces to specific reference people. Important aspects which cannot be described easily but are key in visual appearance. It is a case of unsupervised Learning

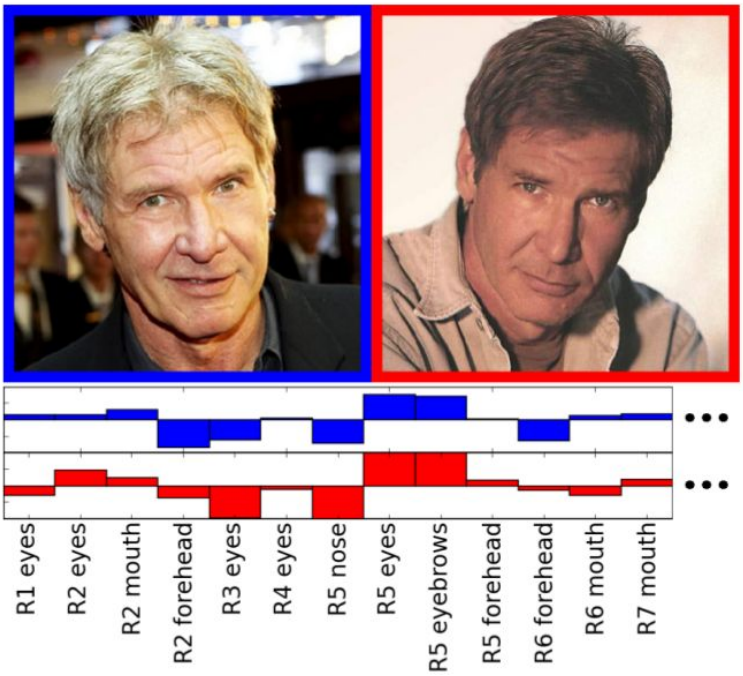
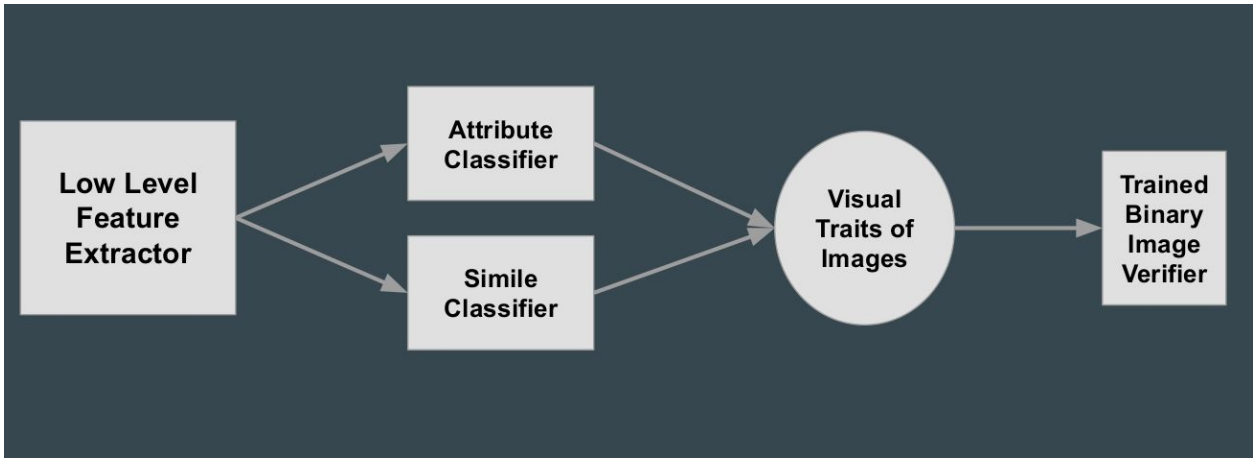


Figure 2 : Simile classifier

PIPELINE



DATASETS USED

1. **Labelled-Faces-In-The-Wild-Dataset** was used for Attribute classifiers, it contains **13233** images of **5749** distinct celebrities. It was split into train/test sets randomly with an 80-20 ratio.

2. **Celebrity-Face-Recognition-Dataset** is an open source effort and is publically available. It Has **800** Images per-person for **99** distinct celebrities. It was Used for **Simile Classifiers.**

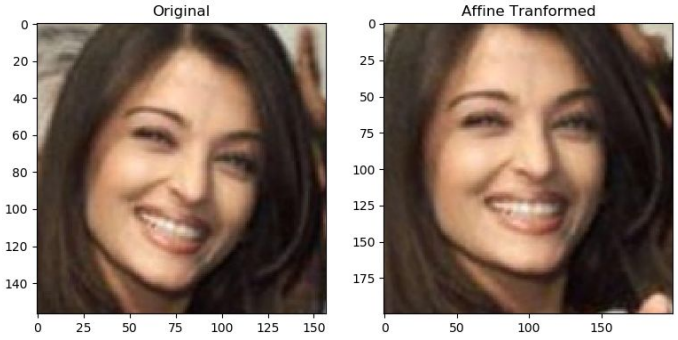
3. The Paper uses **PubFig**-**Dataset** for **Simile Classifiers**, but which could not be used due to copyright and integrity issues.

4. The Final Verification Classifier uses a mix of both the datasets mentioned above.

LOW LEVEL FEATURE EXTRACTION

Our Aim is to provide good meaningful and compact representations of images for training purposes. The method to extract these low-level features are given below -

1. Extract faces from given images using inbuilt CV2 functions.
2. Align all extracted faces to facilitate landmark extraction and also to bring all of them in a common frame of reference for observation.



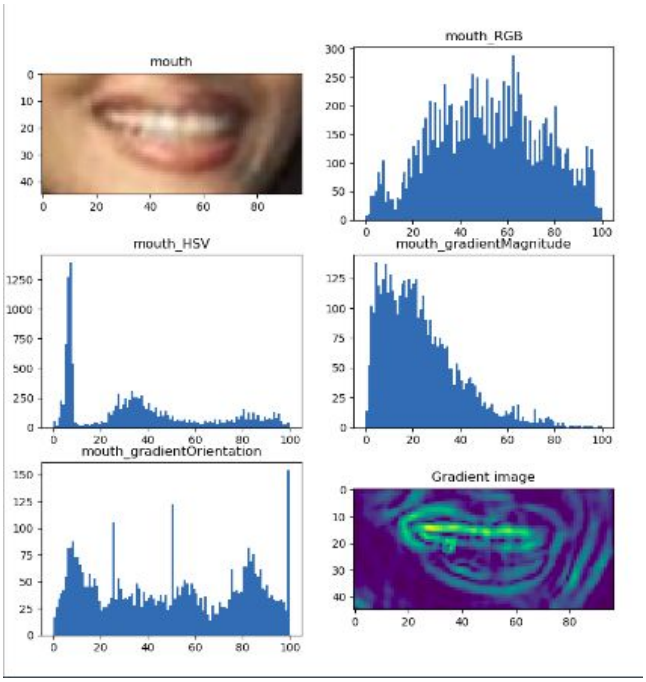
3. Extract important regions from these aligned faces, this helps pinpoint the important regions of the faces which can be used in the representation.
4. This ‘information’ is interpreted and stored in the form of various local/low-level/pixel-level properties of images including :
    - Histograms of RGB,HSV versions of the image.
    - Histograms of gradient magnitude and orientations.
    - Size reduced RGB and gradient image.

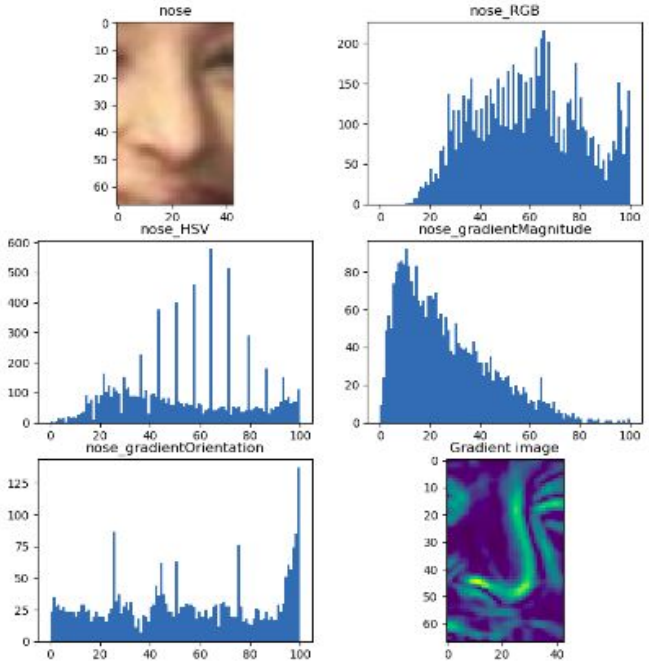
Feature	Normalization	Size
RGB hist / region	(0, 255)	100
HSV hist / region	(Min, Max)	100
Gradient Magnitude hist / region	(Min, Max)	100
Gradient Orientation hist / region	(-pi, pi)	100
RGB image overall	None	50x50
Gradient Magnitude image overall	None	50x50

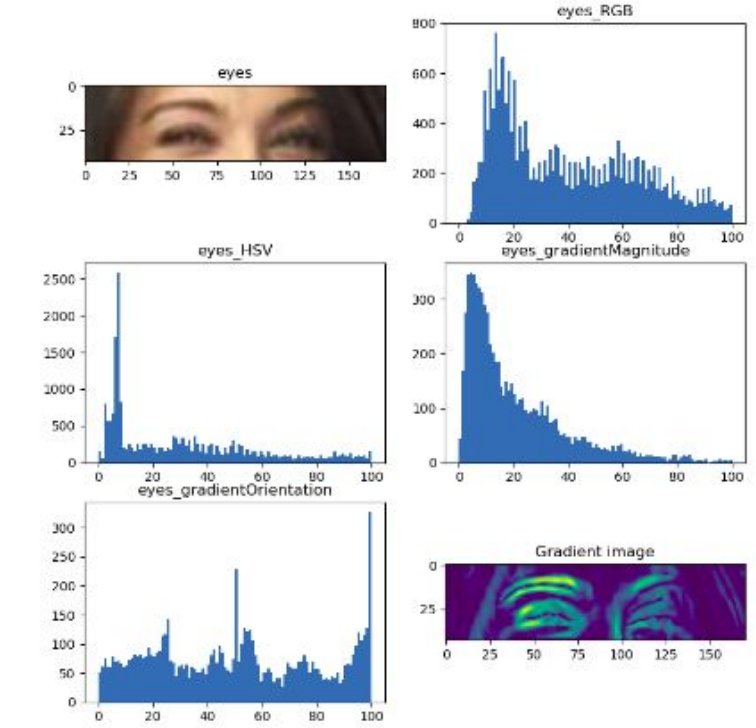
IMPLEMENTATION DETAILS -

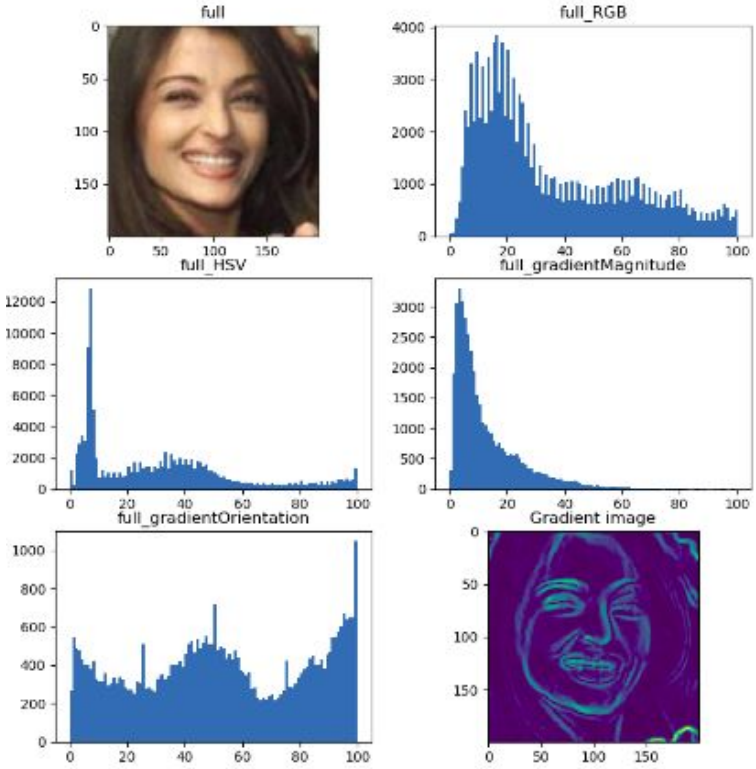
1. HAAR Face detection function available in opencv is used to detect faces.
2. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.
3. To achieve face alignment we constructed affine matrix to transform the image, source and destination points are derived by holding eyes as reference
4. For landmark detection(to extract eyes, nose, mouth) we used inbuilt libraray DLIB, which is a popular face/shape landmark predictor.

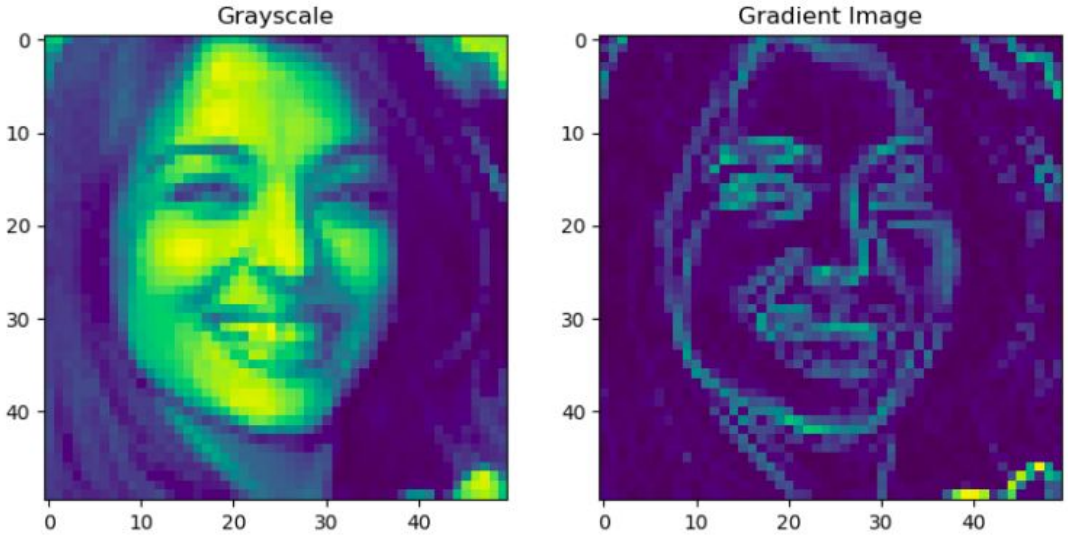
RESULTS -


mouth
mouth_RGB
mouth_HSV
mouth_gradientMagnitude
mouth_gradientOrientation
Gradient image


nose
nose_RGB
nose_HSV
nose_gradientMagnitude
nose_gradientOrientation
Gradient image


eyes
eyes_RGB
eyes_HSV
eyes_gradientMagnitude
eyes_gradientOrientation
Gradient image


full
full_RGB
full_HSV
full_gradientMagnitude
full_gradientOrientation
Gradient image



ATTRIBUTE CLASSIFIER

1 . This is the first type of classifier. For a given attribute, this classifier learns a binary boundary denoting presence or absence of the attribute.
2. The paper mentions 63 predefined attributes like ‘baby’ , ‘Bushy eyebrows’, ‘moustache’ ‘frowning’ . They also have provided annotated values corresponding to each attribute for each image.
3. Also, the paper has mentioned particular pairs and image sets to be used for training so as to facilitate best possible training .
4 .We did achieve good results accuracy-wise(which is the metric used in the paper), but there were some major issues that we need to consider.

IMPLEMENTATION DETAILS -

1. We have used SkLearn’s implementation of grid search to perform SVM classification

2. The paper has mentioned that they used some version of adaboosting for feature selection to reduce the dimensionality, but the authors are not very specific on the method. So for obtaining some initial results we tried out the following two feature sets without feature selection:
    - A concatenation of all the low level features mentioned before.
    - Embeddings generated from a pre-trained neural network.
3. Although some of the traits got pretty good accuracies, on observing the confusion matrices we realised that the datasets were highly imbalanced leading to very poor precision and recall.

RESULTS -

Smiling

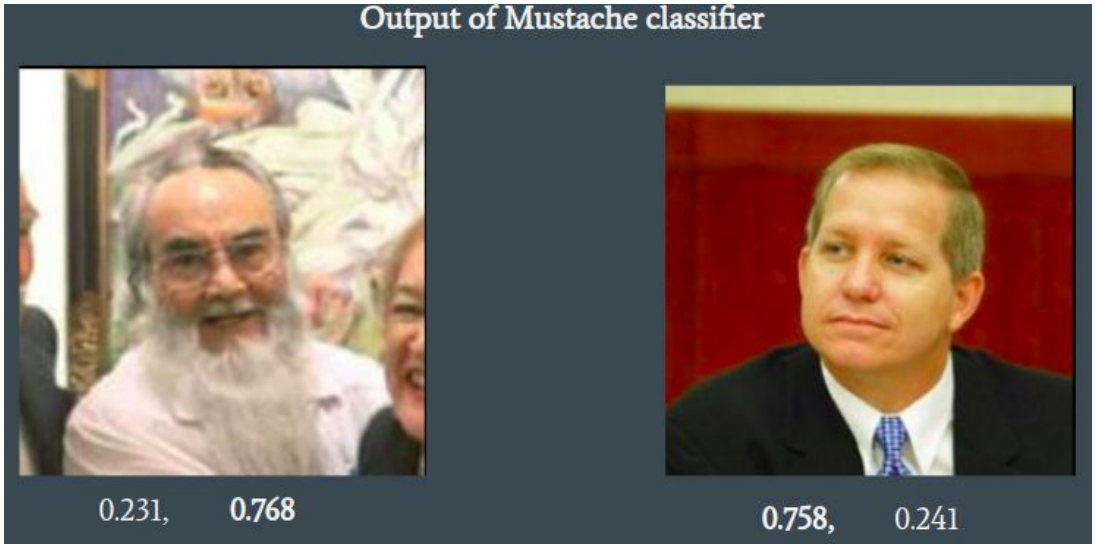
	precision	recall	f1-score	support
False	0.86	0.88	0.87	1642
True	0.83	0.80	0.82	1188
accuracy			0.85	2830
macro avg	0.84	0.84	0.84	2830
weighted avg	0.85	0.85	0.85	2830

Straight Hair

	precision	recall	f1-score	support
False	0.76	0.79	0.77	1631
True	0.70	0.66	0.68	1199
accuracy			0.73	2830
macro avg	0.73	0.72	0.73	2830
weighted avg	0.73	0.73	0.73	2830

High Cheekbones

	precision	recall	f1-score	support
False	0.87	0.86	0.87	1825
True	0.76	0.77	0.76	1005
accuracy			0.83	2830
macro avg	0.81	0.82	0.81	2830
weighted avg	0.83	0.83	0.83	2830



SIMILE CLASSIFIER

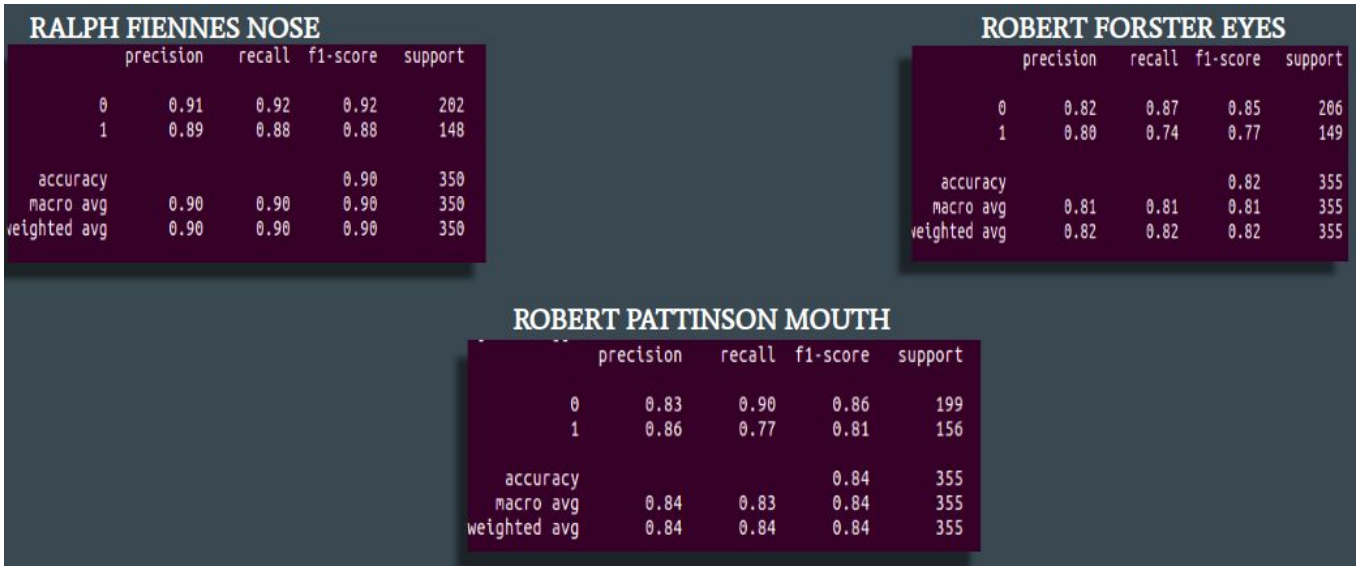
1. The attribute classifiers described in the previous section require each attribute to be describable in words. However, one can imagine that there are many visual cues to people's identities that cannot be described – at least not concisely.
2. The basic idea is that we can describe a person's appearance in terms of the similarity of different parts of their face to a limited set of "reference" people. For example, someone's mouth may be described as similar to Angelina Jolie's, or their nose as similar to Brad Pitt's.

IMPLEMENTATION DETAILS -

1. Aim is to compare train images to specific regions of reference images.
2. Using low level feature extractor described earlier, we extract information of regions like nose, eyes and mouth of both reference and train images.

3. We chose 43 Celebrities and built 3 Classifiers for **[NOSE , MOUTH , EYE]** regions.
4. We built SVM binary classifiers over this, which gave us a total of 129 Simile Classifiers.
5. For Positive samples , we had around 800 positives for each classifier as the celebrity dataset has 800 images of each celebrity.
6. For Negative samples , we used the LFW dataset. We chose 1000 negatives at random as the 2 datasets are disjoint. The paper suggests a higher imbalance, but we found results for this split were better.
7. 80-20 split was used for Training.
8. The SVM classifier was tuned using GridSearch and it uses an RBF kernel.

RESULTS -



FINAL VERIFICATION CLASSIFIER

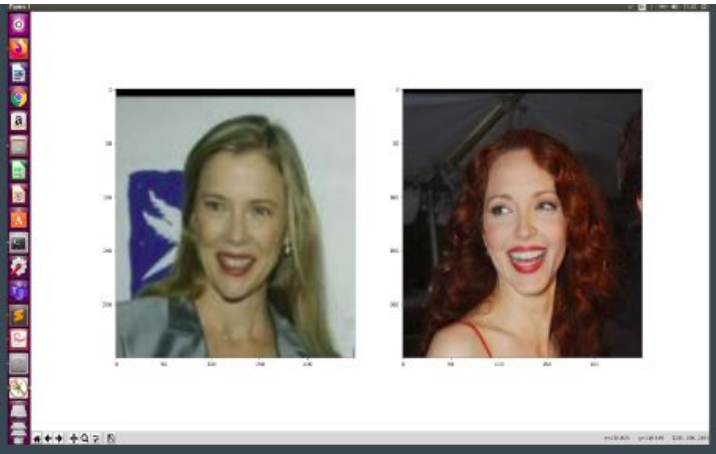
1. In order to make a decision about whether two face images I_1 and I_2 show the same person, we use the final classifier D to compare the trait vectors $C(I_1)$ and $C(I_2)$ obtained by one or both of the methods above.
2. This is the last step of the pipeline
3. Given an input of a pair of images, the classifier identifies whether they belong to the same person.
4. It is a binary SVM Classifier.

IMPLEMENTATION DETAILS -

1. Input is a pair of images. Positive samples are collected from LFW, 2034 Positive pair of images. Negative samples are simply images of two different people, 2000 Negative Samples are collected . Train Test split is 80-20 .
2. Input is preprocessed in the following steps:
    a. For a given image, prepare a vector of the confidence values of all the previously trained classifiers, there are 201 classifiers in total.
    b. After we get representation for both images of an input pair, we combine them using mathematical relations described in the paper to get one single representation vector as input.
3. After this SVM classifier is trained with an RBF Kernel and GridSearch parameter tuning.

RESULTS -

	precision	recall	f1-score	support
0	0.74	0.76	0.75	500
1	0.78	0.75	0.76	547
accuracy			0.76	1047
macro avg	0.76	0.76	0.76	1047
eighted avg	0.76	0.76	0.76	1047

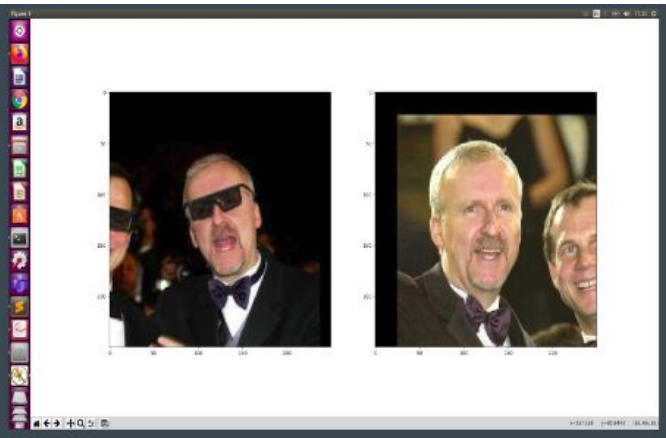


FALSE POSITIVE

- These images do not belong to the same person and are classified as same
- Although even for a human they are similar looking

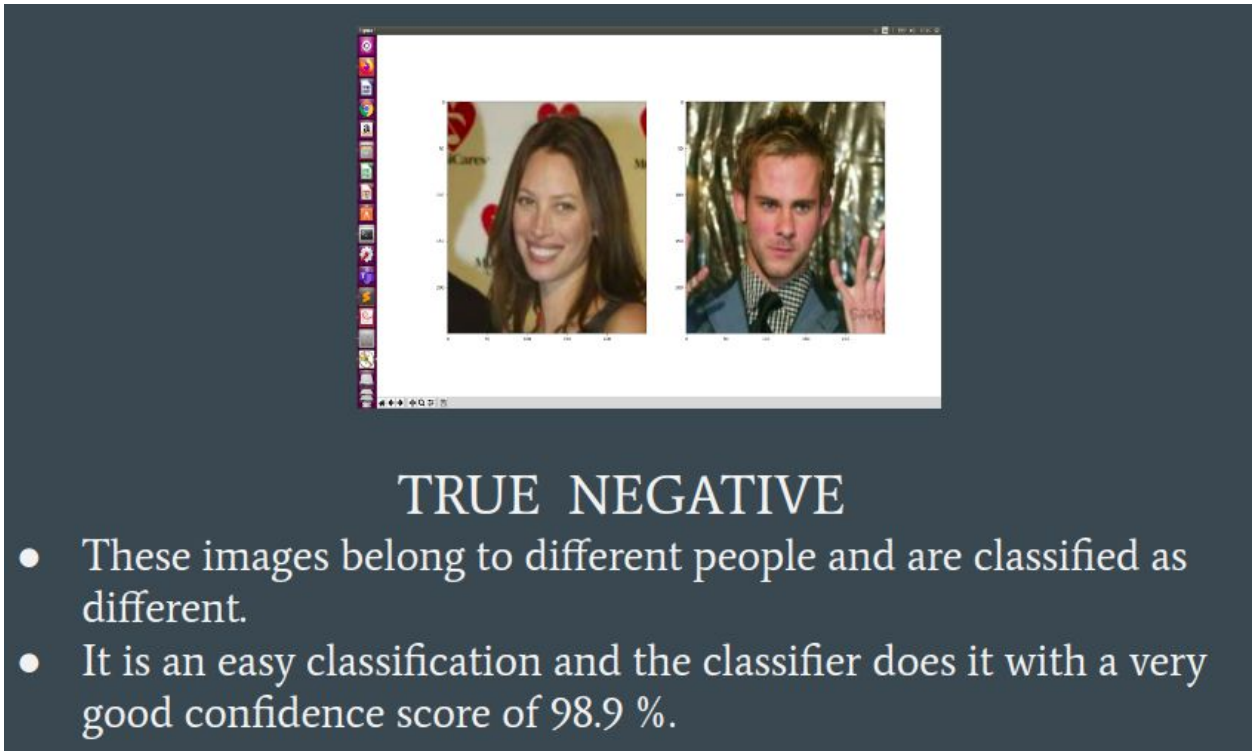
TRUE POSITIVE

- These images belong to the same person and are classified as same.
- It is not a very easy image to classify. The attribute classifiers may be the reason for this good result.



FALSE NEGATIVE

- These images belong to the same person and are classified as different.
- The mistake may be attributed to the sunglasses.



ADDITIONAL EXPERIMENTATION

We also tried training the attribute classifiers with 3 different types of neural networks.

1. Takes the low level features as input. Trained from scratch.

```
NetBN(
  (features): Sequential(
    (0): Linear(in_features=6600, out_features=2048, bias=True)
    (1): BatchNorm1d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Linear(in_features=2048, out_features=512, bias=True)
    (4): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Linear(in_features=512, out_features=64, bias=True)
    (7): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
    (9): Linear(in_features=64, out_features=2, bias=True)
  )
)
```

2. Take the original image as input. Trained from scratch.

```
CNNNet(
  (features): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2))
    (5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2))
    (9): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU()
  )
  (classifier): Sequential(
    (0): Linear(in_features=4608, out_features=1024, bias=True)
    (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Linear(in_features=1024, out_features=128, bias=True)
    (4): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Linear(in_features=128, out_features=2, bias=True)
  )
)
```

3. Takes original image as input. Fine-tuned over imagenet weights.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Feature	SVM	NetBN	CNNNet	Mobilenet v2
Smiling	0.84	0.82	0.87	0.90
White	0.67	0.74	0.76	0.78
Goatee	0.67	0.65	0.65	0.73
Youth	0.64	0.66	0.69	0.75
Mustache	0.65	0.62	0.66	0.76