"EYE SEE YOU"

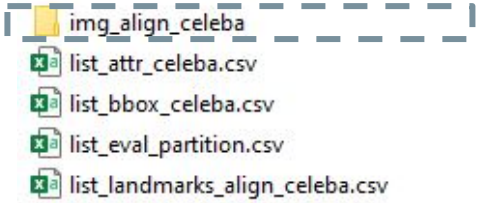
A COMPUTER VISION PROJECT USING THE CELEBRITY ATTRIBUTES KAGGLE DATASET

Classify / Detect using computer vision

Learning Objectives

1. Acquire data and build an image data pipeline
2. Clean and augment image data.
3. Train and deploy a computer vision model.
    1. Image Classification (CNN)
    2. Using Pretrained Models (Viola Jones, VGG16)
    3. Transfer Learning (VGG16 custom--failed)

Data



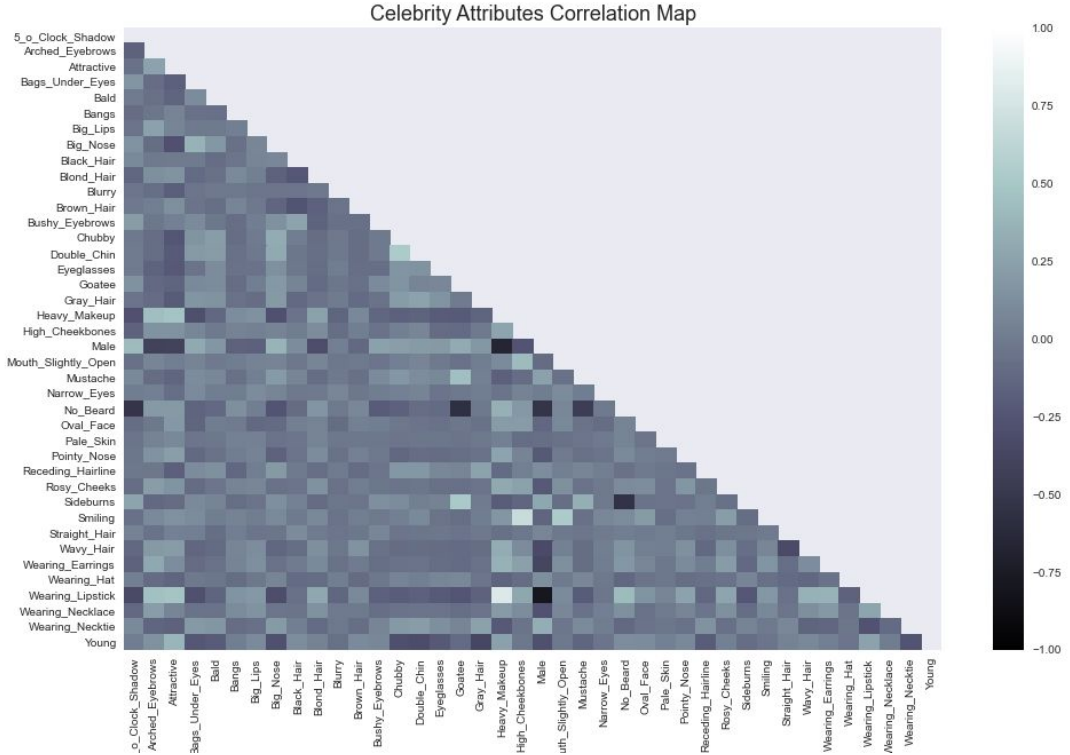

```
Found 202600 files belonging to 1 classes.
```

Source: https://www.kaggle.com/jessicali9530/celeba-dataset

Data

- img_align_celeba
- list_attr_celeba.csv
- list_bbox_celeba.csv
- list_eval_partition.csv
- list_landmarks_align_celeba.csv

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 202599 entries, 0 to 202598
Data columns (total 41 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   image_id            202599 non-null  object
 1   5_o_Clock_Shadow    202599 non-null  int64
 2   Arched_Eyebrows     202599 non-null  int64
 3   Attractive          202599 non-null  int64
 4   Bags_Under_Eyes     202599 non-null  int64
 5   Bald                202599 non-null  int64
 6   Bangs               202599 non-null  int64
 7   Big_Lips            202599 non-null  int64
 8   Big_Nose            202599 non-null  int64
 9   Black_Hair          202599 non-null  int64
 10  Blond_Hair          202599 non-null  int64
 11  Blurry              202599 non-null  int64
 12  Brown_Hair          202599 non-null  int64
 13  Bushy_Eyebrows      202599 non-null  int64
```

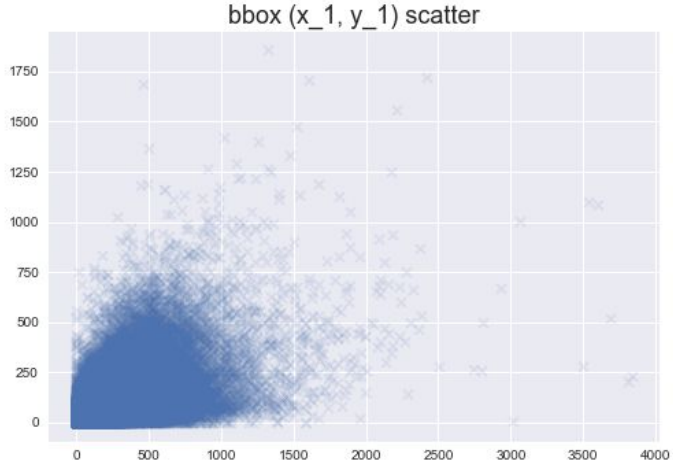


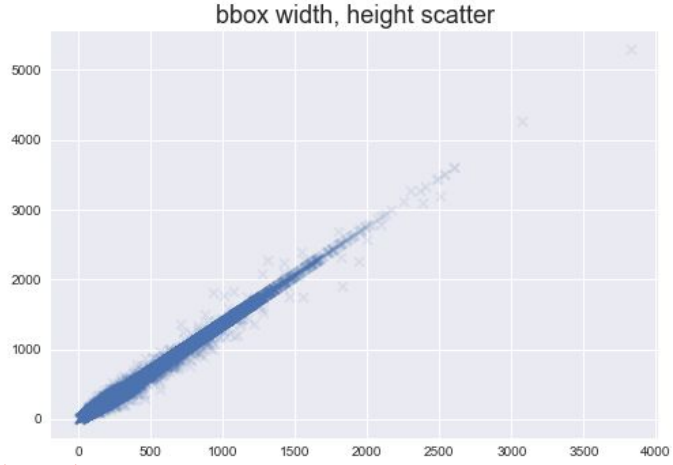
Source: https://www.kaggle.com/jessicali9530/celeba-dataset

Data

- img_align_celeba
- list_attr_celeba.csv
- list_bbox_celeba.csv
- list_eval_partition.csv
- list_landmarks_align_celeba.csv

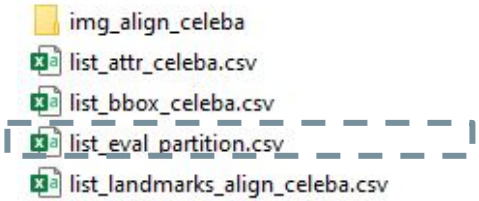
	image_id	x_1	y_1	width	height
0	000001.jpg	95	71	226	313
1	000002.jpg	72	94	221	306
2	000003.jpg	216	59	91	126

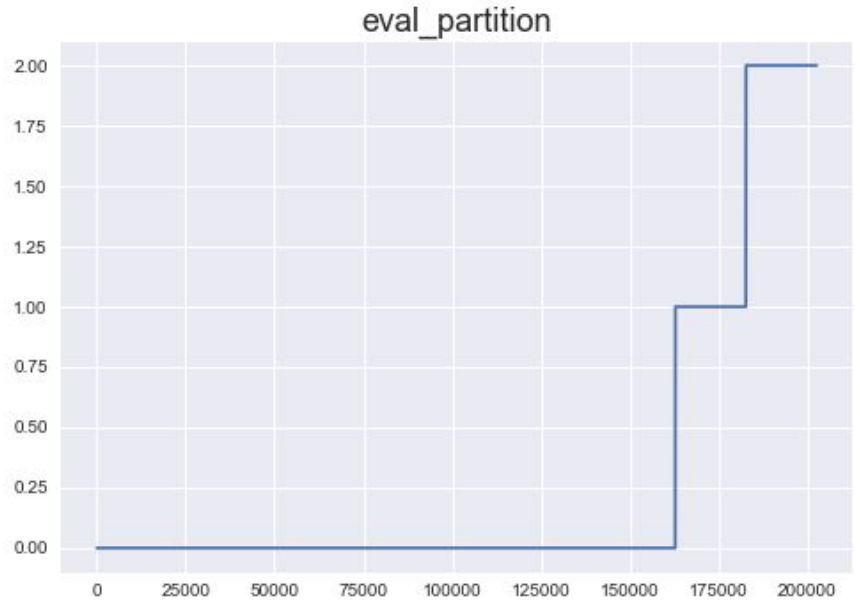




Source: https://www.kaggle.com/jessicali9530/celeba-dataset

Data



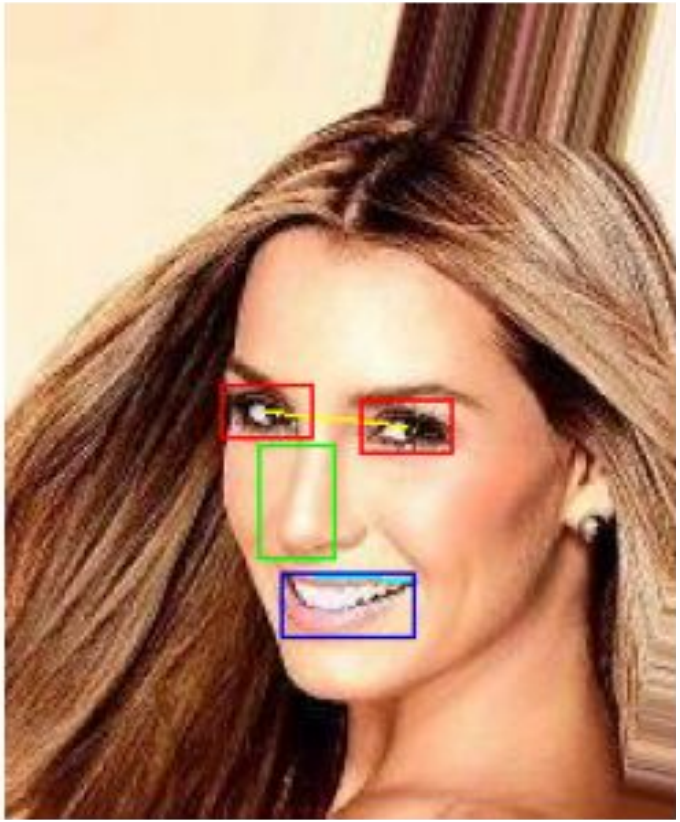


Source: https://www.kaggle.com/jessicali9530/celeba-dataset

Data

- img_align_celeba
- list_attr_celeba.csv
- list_bbox_celeba.csv
- list_eval_partition.csv
- list_landmarks_align_celeba.csv

	image_id	lefteye_x	lefteye_y	righteye_x	righteye_y	nos
0	000001.jpg	69	109	106	113	
1	000002.jpg	69	110	107	112	
2	000003.jpg	76	112	104	106	



Source: https://www.kaggle.com/jessicali9530/celeba-dataset

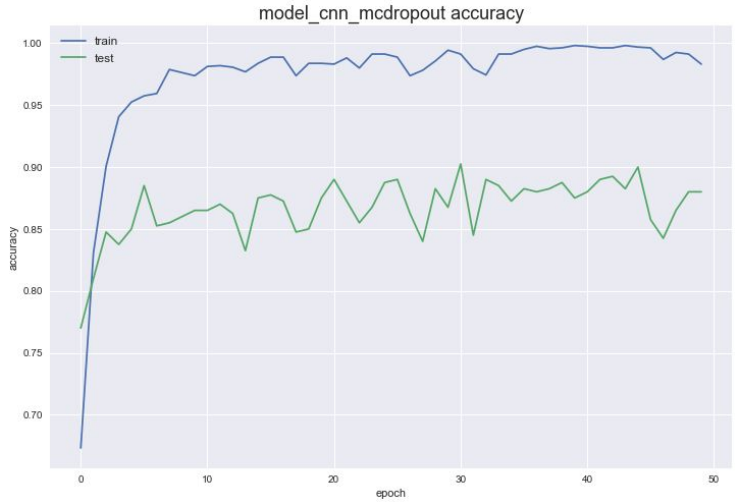
Model (CNN; left/right eye)

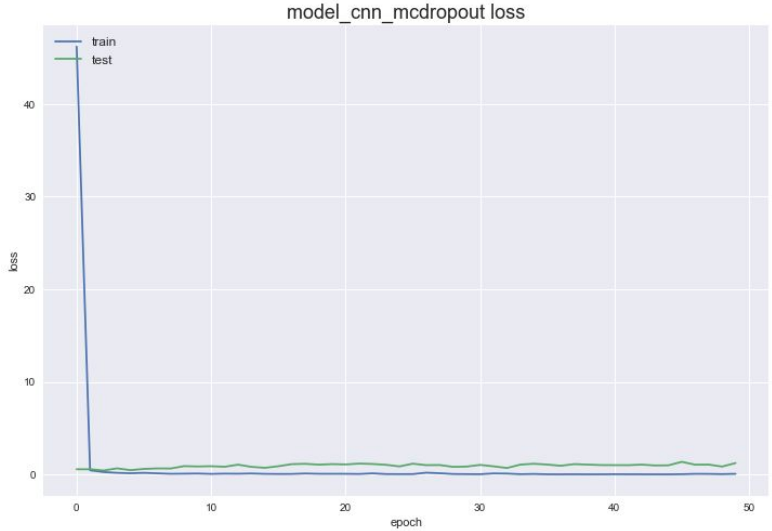
```
model_cnn_mcdropout.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 30, 30, 8)         224
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 28, 28, 16)        1168
_________________________________________________________________
flatten_3 (Flatten)          (None, 12544)             0
_________________________________________________________________
mc_dropout_9 (MCDropout)     (None, 12544)             0
_________________________________________________________________
dense_9 (Dense)              (None, 256)               3211520
_________________________________________________________________
mc_dropout_10 (MCDropout)    (None, 256)               0
_________________________________________________________________
dense_10 (Dense)             (None, 256)               65792
_________________________________________________________________
mc_dropout_11 (MCDropout)    (None, 256)               0
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 514
=================================================================
Total params: 3,279,218
Trainable params: 3,279,218
Non-trainable params: 0
_________________________________________________________________
```

```
class MonteCarloDropout(keras.layers.Dropout):
  def call(self, inputs):
    return super().call(inputs, training=True)
```

More on Monte Carlo Dropout: https://arxiv.org/abs/1506.02142
Also: https://towardsdatascience.com/monte-carlo-dropout-7fd52f8b6571

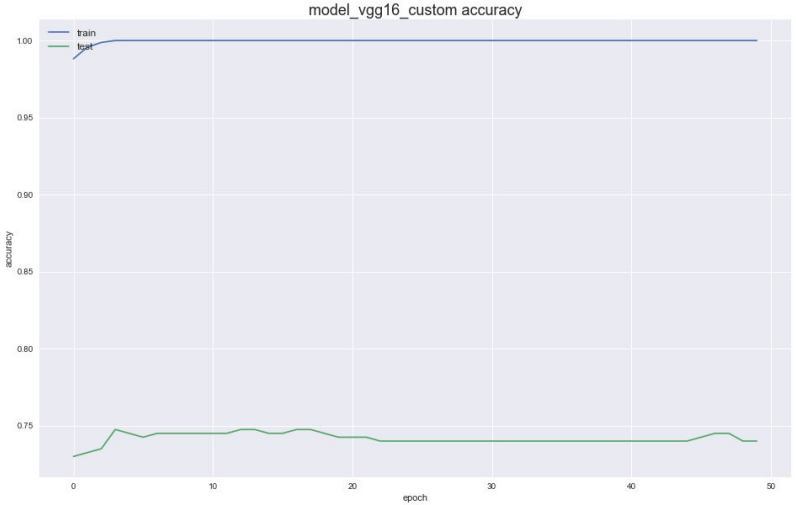


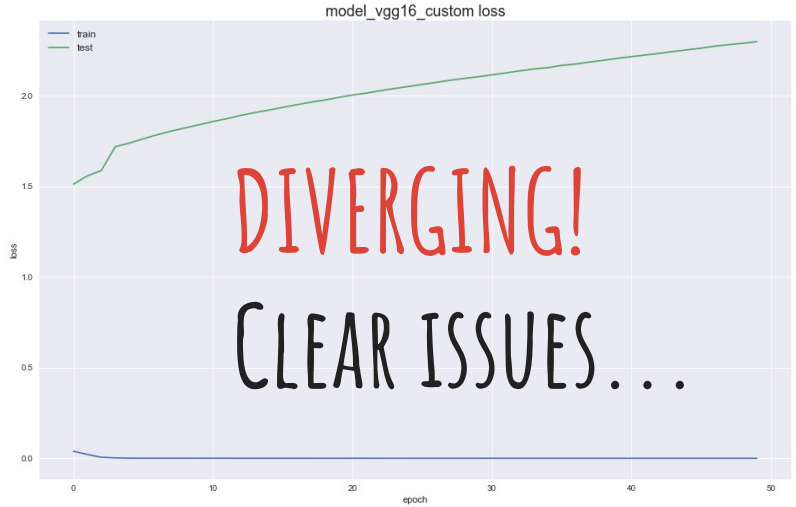


Model (VGG16 Custom)

```
1 model_vgg16_custom.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 1, 1, 512)         14714688

 global_average_pooling2d (G  (None, 512)              0
 lobalAveragePooling2D)

 flatten_9 (Flatten)         (None, 512)               0

 dense1 (Dense)              (None, 256)               131328

 dense2 (Dense)              (None, 10)                2570

 dense_16 (Dense)            (None, 2)                 22

=================================================================
Total params: 14,848,608
Trainable params: 133,920
Non-trainable params: 14,714,688
_________________________________________________________________
```

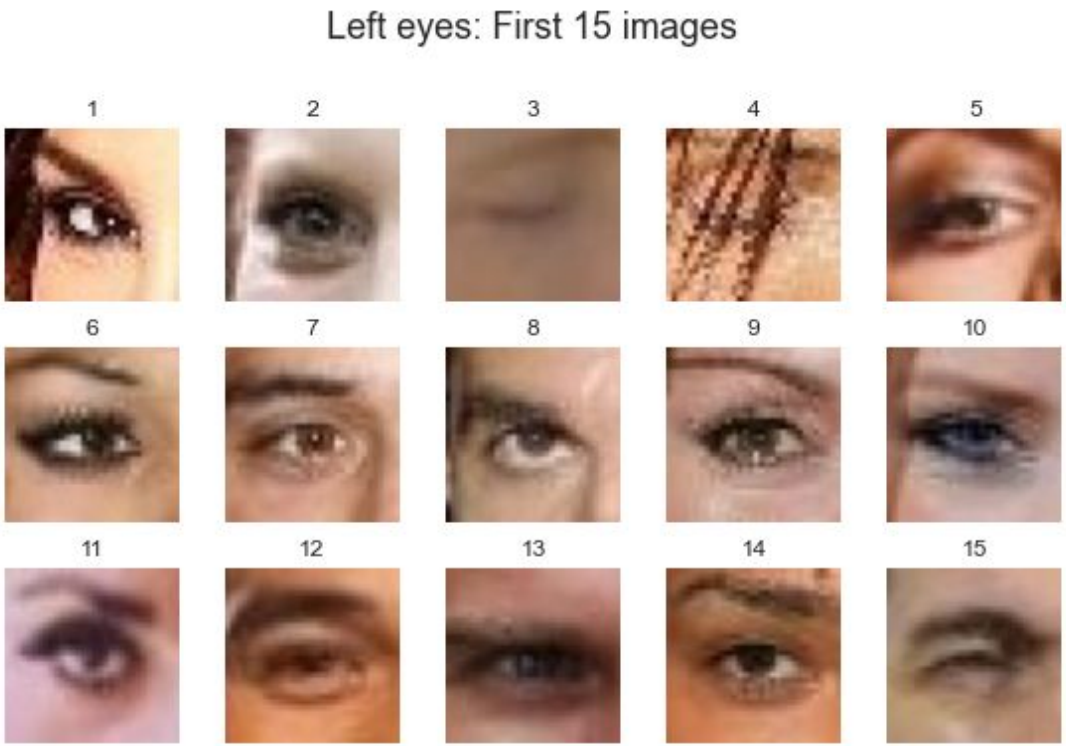


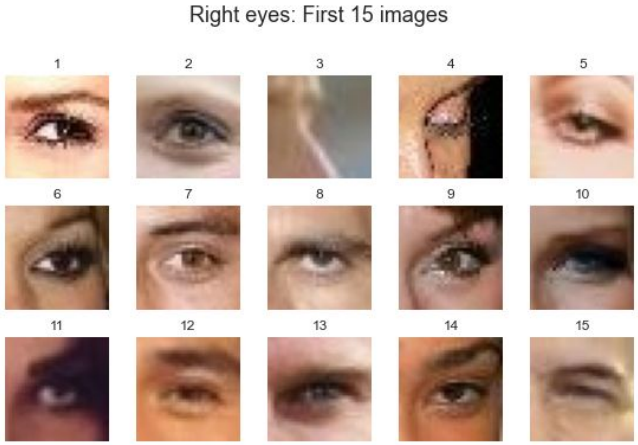


DIVERGING!

Clear issues...

Cropped Images (1000 left eyes, 1000 right eyes)



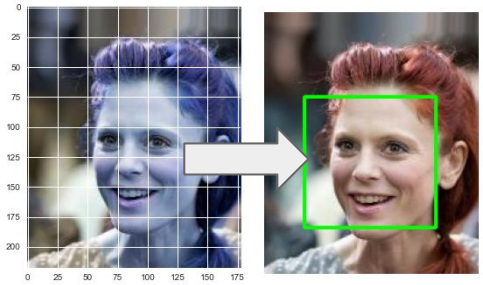


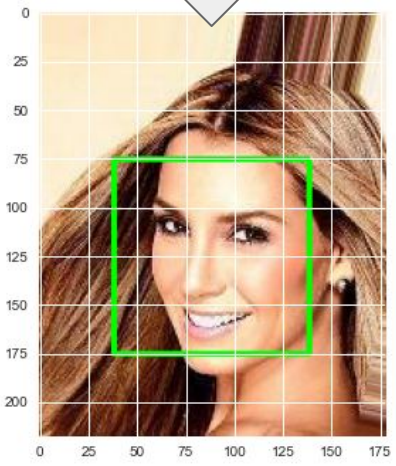
Frontal Face Detection (Viola-Jones / Haar Cascade)

1. Read in image (gray)
2. Convert to Integral Image; “Constant time” / O(1)
3. Attention operator; Haar Cascade

```
===== Test Image, <class 'numpy.ndarray'> =====
[[233 233 233 ... 232 241 241]
 [233 233 233 ... 234 241 241]
 [233 233 233 ... 236 241 242]
 ...
 [ 88  63  93 ...  72  73  73]
 [ 77  85 113 ...  66  68  68]
 [115 151 192 ...  66  68  68]]
===== Shape =====
(218, 178)
```

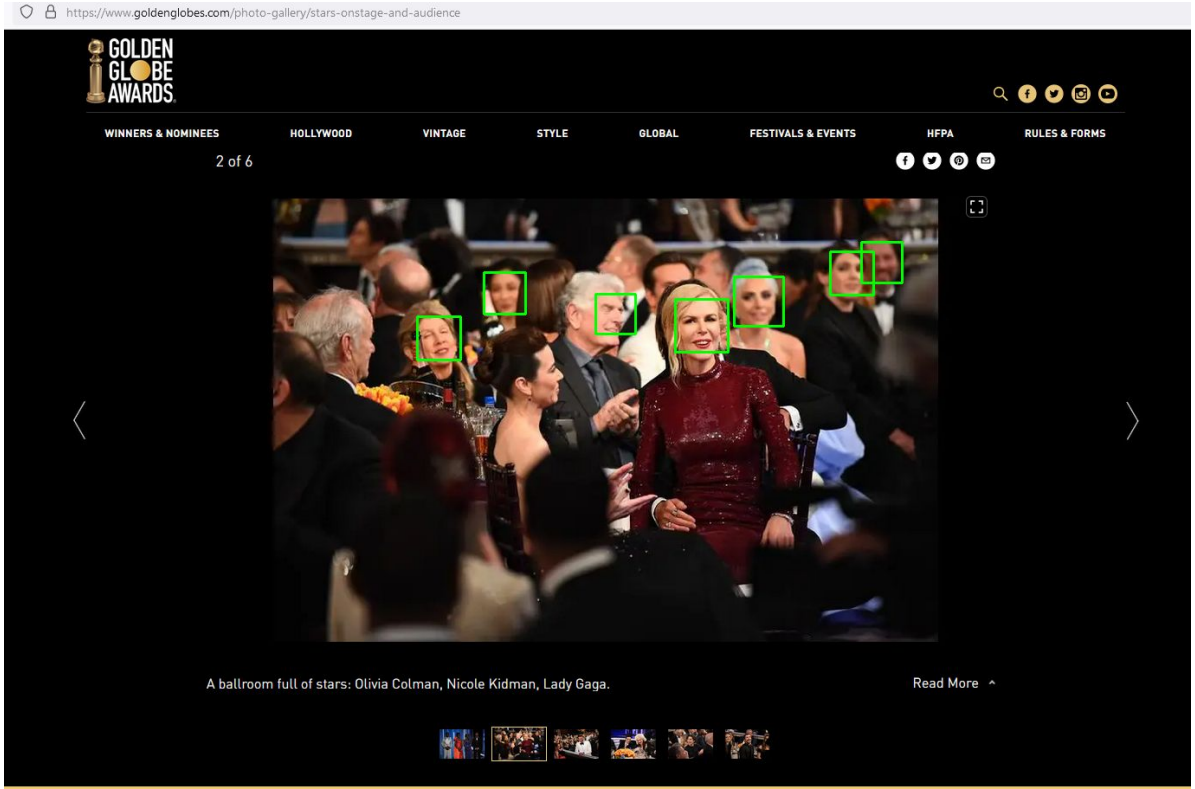
```
(219, 179)

array([[      0,       0,       0, ...,       0,       0,       0],
       [      0,     233,     466, ...,   31180,   31421,   31662],
       [      0,     466,     932, ...,   62286,   62768,   63250],
       ...,
       [      0,   38151,   75925, ..., 5598109, 5636022, 5674017],
       [      0,   38228,   76087, ..., 5614686, 5652667, 5690730],
       [      0,   38343,   76353, ..., 5631455, 5669504, 5707635]],
      dtype=int32)
```





Application to Test Image

Very fast detection (less than a second) of all relevant frontal faces in a test image.



Blurring faces

Using the face detection mechanism, filters like blurs can be easily added to the image.