# INTRODUCTION

Object detection is a computer technology in the field of computer vision which is used to identify objects in a digital image. Face detection is a type of object detection which is used to identify the location and size of the frontal human faces within an image.

Some of the leading applications of face detection comprise of

* **Facial motion capture**: converts the movement of a person’s face into a digital database which can be used for animations, movies, games.
* **Facial recognition**: used to identify or verify a person from digital image or a video frame.
* **Photography**: digital cameras in today’s world uses autofocus to detect faces. Additionally, smile detection can be used for taking photo at an appropriate time.
* **Emotional inference**: It is used to help people with autism to understand the feelings of people around them.

The libraries used in this project are:

* **OpenCV**: OpenCV is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.
* **DLIB**: Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems such as face detection, face recognition, etc.

# DATASET

WIDER FACE dataset is a face detection benchmark dataset, of which images are selected from the publicly available WIDERFACE dataset. We choose 32,203 images and label 393,703 faces with a high degree of variability in scale, pose and occlusion as depicted in the sample images. WIDER FACE dataset is organized based on 61 classes. Each class is describing an event. For each event class, we randomly select 40%/10%/50% data as training, validation and testing sets. A face detector is trained using WIDER FACE training/validation partitions and tested on WIDER FACE test partition.

* WIDER Face Training Images (12880 images)
* WIDER Face Validation Images (3226 images)
* WIDER Face Testing Images (16097 images)
* Face Annotations

The dataset is not limited to the fixed size images. It consists of images with different size and scale.

The annotation contains the ground truth values consists of Number of bounding boxes, x-coordinate, y-coordinate, width, height, blur, expression, illumination, invalid, occlusion and pose.

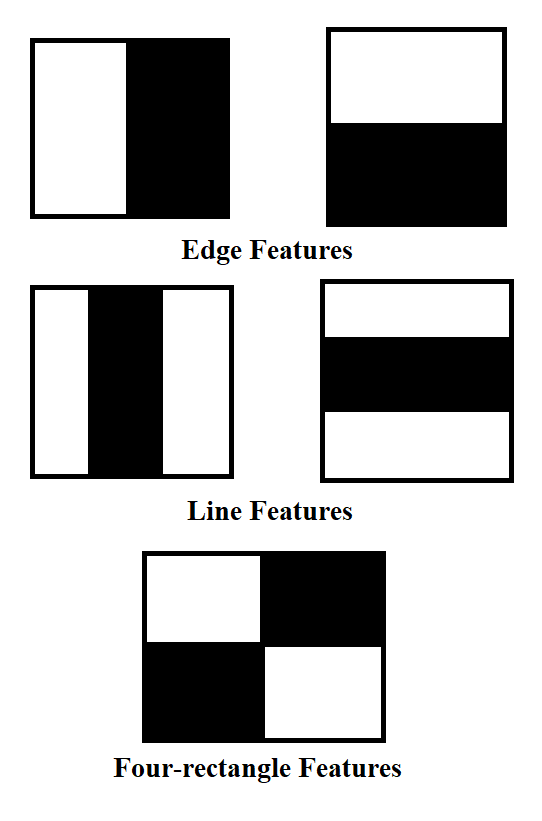
* Blur is categorized into 3 category clear, normal blur and heavy blur.
* Expressions are categorized into typical expression and exaggerate expression.
* For Occlusion, a face is defined as ‘no occlusion’ for 0% as ‘partially occluded’ for 1%-30% and as ‘heavily occluded’ if over 30% of the total face area is occluded.
* Pose is defined as two pose deformation levels, namely typical and atypical. we assign a face pose as atypical if either the roll or pitch degree is larger than 30-degree; or the yaw is larger than 90-degree. Otherwise a face pose is classified as typical.
* Illumination also categorized into normal illumination and extreme illumination.

# MODELS

**Haar Cascade Classifiers**

Face Detection using Haar cascade classifiers is a machine learning based approach in which a cascade function is trained from a lot of positive and negative images. Instead of using strong classifier, concatenation of several weak classifier is called cascade classifier is preferred. Positive images are the images with faces in it and the negative images are the images without faces.

The algorithm needs a lot of positive images and negative images to train the classifier to extract features from it. Haar features are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.



All possible sizes and locations of each kernel is used to calculate several features. For each feature calculation, we need to compute sum of pixels under white and black rectangles.

We need to implement each feature on all the training images. And for each feature, find the best threshold which will classify the faces to positive and negative. We will select the features with the lowest error rate, in order to find the features that best classifies the face and non-face images.

**Histogram of Oriented Gradients**

HOG works like a sliding window. A block is considered as a pixel grid in which gradients are constituted from the magnitude and direction of change in the intensities of the pixel within the block.

The idea behind HOG is to extract features into a vector and feed it into a classification algorithm like a Support Vector Machine for example that will assess whether a face (or any object you train it to recognize actually) is present in a region or not.

In the HOG feature descriptor, the distribution (histograms) of directions of gradients (oriented gradients) are used as features. Gradients (x and y derivatives) are typically large around edges and corners and allow us to detect those regions.

The process was implemented for human body detection, and the detection chain was the following:

* Pre-processing: The input images must be of the same size (crop and rescale images).
* Calculate the Gradient Images: Compute the horizontal and vertical gradients of the image, by applying kernels. The gradient of an image typically removes non-essential information.
* Calculate Histogram of Gradients: The image is then divided into 8x8 cells to offer a compact representation and make our HOG more robust to noise. Then, we compute a HOG for each of those cells.
* Block Normalization: Normalize the image and make it invariant to lighting by constructing 16x16 block. This is simply achieved by dividing each value of the HOG of size 8x8 by the L2-norm of the HOG of the 16x16 block.
* Calculate the HOG feature vector: To calculate the final feature vector for the entire image patch, vectors are concatenated into one giant vector.

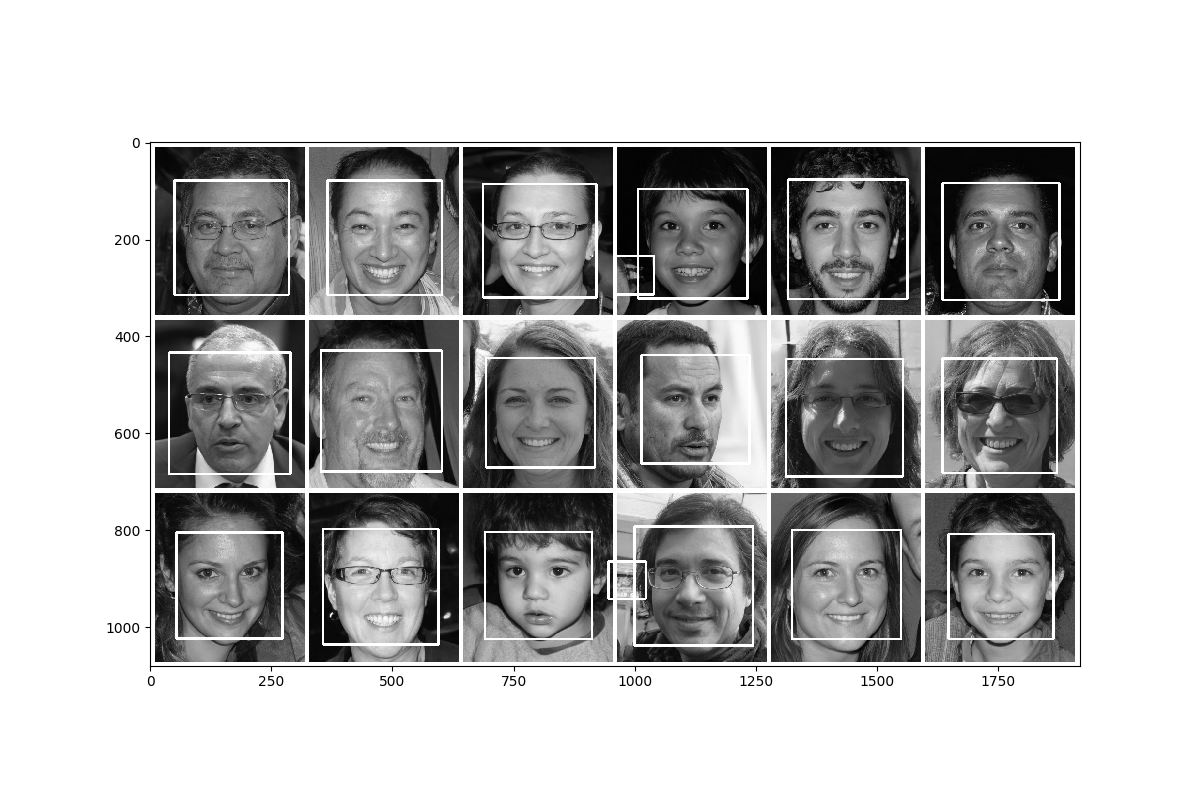
**Convolutional Neural Networks**

One of the most popular models used in computer vision is CNN which are a feed-forward neural network. CNN’s offer an automated image pre-treatment as well as dense neural network part. CNN are majorly used for processing data with grid-like topology.

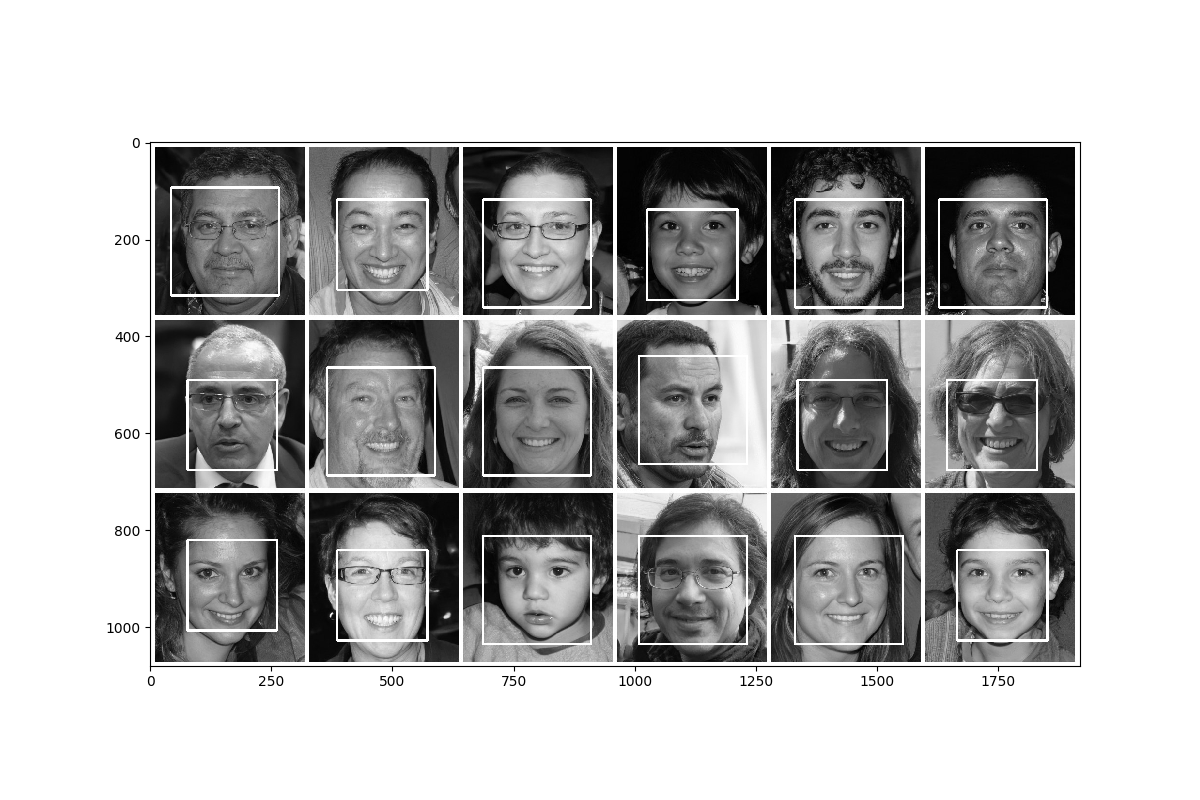
With the rise of deep learning and greater computation capabilities, this work of extracting features in order to extract as much information from the image as possible can now be automated. The model makes use of the pre-trained model defined in mmod\_human\_face\_detector.dat. This pre-trained model was created based on a dataset containing face images from ImageNet, AFLW, Pascal VOC, the VGG dataset, WIDER, and face scrub.

# EXPERIMENT

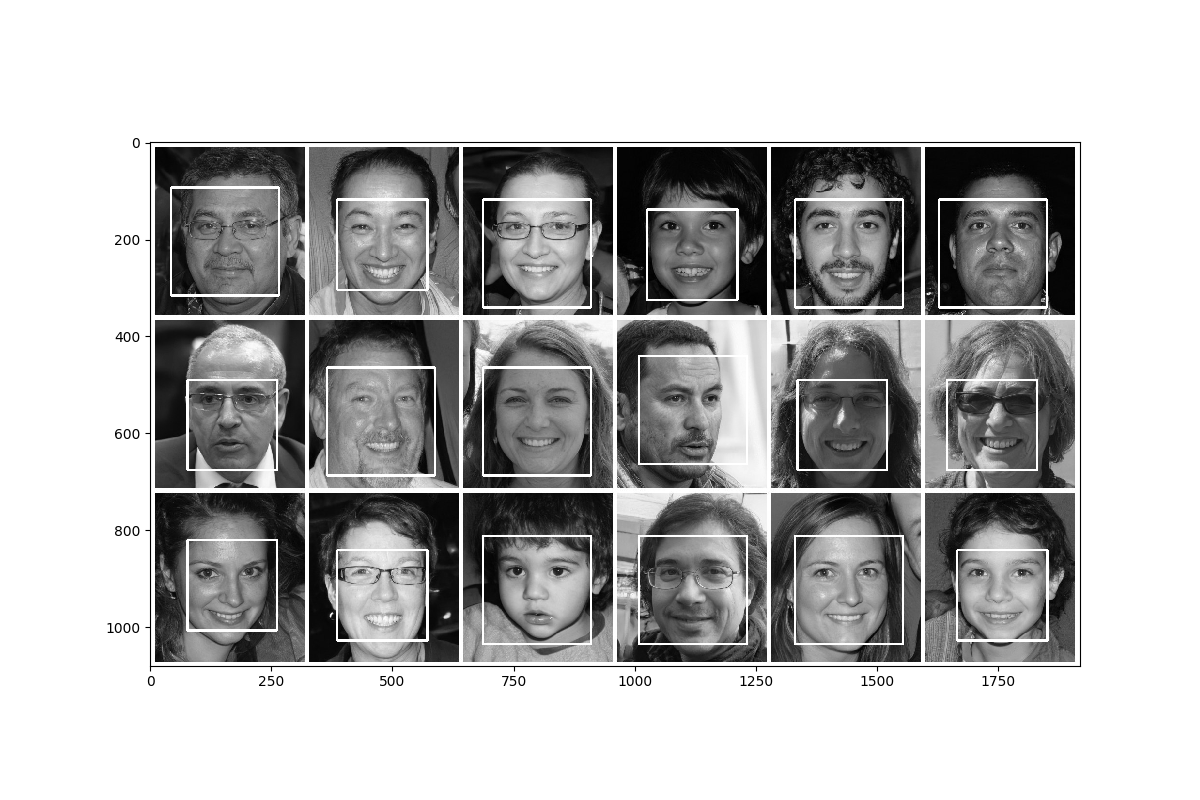
The data from WIDER Face Validation set was filtered to omit difficult test cases that resulted in extremely low true positive percentages. These include images with too much blur (the actual quantitative measure for blur is given on a scale of 0 to 2, the pruning steps omitted bounding boxes with blur values and occlusion values of 2) or faces whose bounding boxes were less than 35 x 35 pixels. Also, images that were greater than 950 pixels in height (all images had a width of 1080 pixels) were also pruned to speed up computation and prevent the CNN model from crashing on the PC it was running on. The original validation dataset of 3226 images was reduced to 1667 images and a total of 5543 ground truth bounding boxes. Each of the three models was run on this data set and their correctness was tested against the ground truth. All experiments were performed on an Linux (Ubuntu) laptop with an intel CORE i5 CPU.

  
Illustration 1: Output for Cascade Model on a Test Image. Note the false positive bounding boxes in the first and last row.

asdf


  
Illustration 2: Output for CNN Model on a Test Image

Each model returned a list of bounding boxes for each image. The bounding boxes for that image were compared with the ground truth bounding boxes by computing their confidence for each box using Intersection Over Union (IOU).

  
Illustration 3: Output for HoG Model on a Test Image.

# INTERSECTION OVER UNION

To compute the confidence of a match with ground truth we first find the overlapping rectangle (see code for how this is done). Next we compute the area of the intersection of the overlap in the following way: Recall that a bounding box (rectangle) can be defined as a pair of (x,y) coordinates one for top left and one for bottom right. Let **a** and **b** be the two rectangles and let (x1, y1) and (x2, y2) be the pair of coordinates. Then the intersecting bounding box **I** can be computed in the following way: **I**x1 **= max(a**x1, **b**x1),  **Iy**1 **= max(a**y1, **b**y1), **I**x2 **= min(a**x2, **b**x2), and  **I**y2 **= min(a**y2, **b**y2). This method can be proven correct by simple inspection of the any pair of overlapping rectangles. Next we use Principle of Inclusion-Exclusion to determine the union of the area of rectangles which is **area(U) = area(a)** + **area(b)** – **area(I)**. Next we can compute the IOU by dividing the union from the intersection (**area(I)/area(U)**) to determine the percentage of overlap. If the overlap percentage was greater than or equals to a threshold, it was accepted as a true-positive and false-positive otherwise. The threshold value for our experiment was set to 0.50.

# RESULTS

After conducting our experiments, we had the below results for each model:

***CASCADE (1st out of 2 runs)***

*Computation Time: 3168.35 seconds (~52.8 minutes)*

|  |  |
| --- | --- |
| CASCADE | |
| Blur | 0.197192 |
| Expr | 0.212446 |
| Illum | 0.206818 |
| Occlu | 0.069542 |

Having a higher true-positive rate is “good” in the sense that the model is correct more often, however this can be a poor descriptor of performance by the following example: output a multitude of possibly true bounding boxes all over the image to increase the true-positive percentage. So we also consider the precision (true-positive over true-positive + false-positive). The ultimate goal in facial detection (or any object detection) is to minimize the number of false-negatives (ie. there was a face in the image, but the model did not detect it). The tables represent the true-positive ratio for ground truth bounding boxes with some amount of blur, expression, illumination variation, and occlusion.

***HOG***

*Computation Time: 606.80 seconds (~10.1 minutes)*

|  |  |
| --- | --- |
| HOG | |
| Blur | 0.211996 |
| Expr | 0.242489 |
| Illum | 0.222727 |
| Occlu | 0.065141 |

The HoG model’s performance is a tremendous improvement from the Cascade model in a multitude of ways. In addition to the obvious improvement to the true positive rate, its precision is also much higher (fewer false-positives, 743). The number of false-negatives is also reduced meaning HoG detected more faces when they were not detected in Cascade. All this was done in 606.8 seconds which is almost 1/5th the time for Cascade.

***CNN***

*Computation Time: 70192.76 seconds (~19.5 hours)*

|  |  |
| --- | --- |
| CNN | |
| Blur | 0.330891 |
| Expr | 0.334764 |
| Illum | 0.329545 |
| Occlu | 0.191021 |

The CNN is yet another improvement from the previous models. CNN raises the bar again by having a raising the true positive rate by 25 percent, reducing the number of false-negatives and reducing the number of false-positives ie CNN makes more correct bounding boxes, makes fewer incorrect bounding boxes, and detects faces that were too ‘difficult’ or obscure to find by the other models. Using the CNN, though, comes with the consequence of requiring huge computational power. The model required almost 20 hours of computation time on a core i5 processor. This model requires a GPU to perform in real time to be practical. In all, CNN is the most robust, but expensive, model.

**CASCADE (2nd run out of 2 runs)**

*Computation Time: 1666.10 seconds (~27.77 minutes)*

|  |  |
| --- | --- |
| CASCADE | |
| Blur | 0.220238 |
| Expr | 0.223176 |
| Illum | 0.230682 |
| Occlu | 0.090669 |

The cascade model was run for a second time with parameter adjustments. The scale factor was reduced to 1.05 which means the image is down-scaled by 5% on each iteration which increases the chance of matching a face. This is different from the 1.10 (10% down-scaling rate) that was used in the first run of the cascade model. The trade-off for this is the increased number of iterations to find a match and therefore total speed.

# CONCLUSION

In this experiment we compared and contrasted the results of three face detection models—Cascade classifiers, Histogram of Oriented Gradients and Convolutional Neural Networks. The true-positive, precision and false-discovery rates were compared with each of the models.

The upsides of Cascade is its computational speed. While not as fast as HoG, it is still much more practical to use than a CNN requiring less power. While Cascade produced the worst results, the model can be tuned to perform better by decreasing the scale factor and thus improving the detection rate. Also, Cascade requires no offline training of SVM or neural nets like HoG or CNNs do. They are more portable and ‘lightweight’ than the other two models. The downsides of Cascade are its accuracy and precision which were the worst of the three models. The upsides of HoG are also its speed combined with its accuracy and precision. The downsides of HoG are it is still not quite up to par with a CNN’s accuracy. The upsides of the CNN are its robustness. CNN had the best results in terms of detection and in all categories of attributes (blur, illum., etc.). The downsides of CNN is requiring more computational power (GPU).

In conclusion, the clear ‘winner’ of the three models is Histogram of Oriented Gradients. It is the most practical to use for a simple face detection. It’s fast and relatively accurate. HoG offers the best trade-off between computation and correctness. However, when robustness is needed and computational power is provided, CNN is the best model for face detection. If not, then Cascade is still a better model than CNN because of its lightweight and its speed which can be increased by increasing the scale factor.

All source code and dataset images can be found on our github page: <https://github.com/SDSUCV696/project>

# BIBLIOGRAPHY

[1]: R. Padilla, C. F. F. Costa Filho and M. G. F. Costa. Evaluation of Haar Cascade Classifiers Designed for Face Detection.

# [2]: Josephine Julina J.K. T.Sree Sharmila. Facial recognition using histogram of gradients and support vector machines. IEEE Conference 2017

# [3]: WIDERFACE Dataset: <http://shuoyang1213.me/WIDERFACE/>

[5]:<http://dlib.net/dnn_mmod_ex.cpp.html>

[6]:<https://www.learnopencv.com/histogram-of-oriented-gradients/>