The Hong Kong Polytechnic University
Department of Electronic and Information Engineering

EIE4430 / EIE4433 Honours Project

1. Student Name: _____Leung Siu Ngai_____ (Student No.: 16060903D)

2. Programme Code: 42470

3. Project Title: Support Vector Machine in Supervised Learning

4. Supervisor Name: Dr. Liang LIU

5. Project summary (State clearly the project objectives and results achieved by yourself.)

[Please list in point form where appropriate]

Objectives

1. Learn about the SVM through literature study and online research.
2. Implement a practical application using the SVM classifier, and the application is decided to be face detection.
3. Collect the experiment results and study the results.

The face detection accuracy can up to 100% if the image faces are directly facing to the camera. It also depends on the face orientations. If the face tilts or turn to the side, the detector may occasionally detect the face. Overall, the average detection accuracy can still up to 80~85%.

DECLARATION OF ORIGINALITY

Except where reference is made in the text of this report, I declare that this report contains no material published elsewhere or extracted in whole or in part from any works or assignments presented by me or any other parties for another subject. In addition, it has not been submitted for the award of any other degree or diploma in any other tertiary institution.

No other person's work has been used without due acknowledgement in the main text of the Report.

I fully understand that any discrepancy from the above statements will constitute a case of plagiarism and be subject to the associated.

Leung Siu Ngai

Signature

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

DEPARTMENT OF ELECTRONIC AND INFORMATION ENGINEERING

電子及資訊工程學系

Department of Electronic and Information Engineering

Final Year Project Final Report (2020/21)

Support Vector Machine in Supervised Learning

Student Name: Leung Siu Ngai

Student ID: 16060903D

Programme Code: 42470

Supervisor(s): Dr. Liang LIU

Submission Date: 22 April 2021

1. Index

2. Abstract

This report will first introduce some background information about Support Vector Machine (SVM). Later, a face detection application will be developed using the SVM classifier and Histogram of Gradients feature descriptor. The primary task of phase II project is to create a face detector. Hence this report will be more focused on the methodology part. The approach used to do the detection is simple and easy to implement, which is using a sliding window to scan through every pixel to seek the faces in an image. The SVM kernel used for the face detector will be linear only. Unlike the state-of-art Viola-Jones face detector, giving quick and accurate detection, the sliding window approach may produce many redundant detections. In the experiment part, the results of detections will be list in detail.

3. Objective

In this project, there will be three objectives:

4. Learn about the SVM through literature study and online research.
5. Implement a practical application using the SVM classifier, and the application is decided to be face detection.
6. Collect the experiment results and study the results.

Then use the data get in the experiment to complete this final report. Also, the application will be tuned in the best condition such that the demonstration of this project can run smoothly.

4. Background

Support Vector Machines (SVMs), in 1992, was introduced by Boser, Guyon, and Vapnik, mainly handle classification and regression types of problem by using a series of related supervised learning approaches [1]. More precisely speaking, SVM is a tool that takes advantage of the machine learning mechanism to maximize predictive accuracy. Meanwhile, prevent over-fit to the data [1].

5. Introduction of Support Vector Machine (SVM)

Before deep into the topic, we may first consider a case, a simple linear classification problem, as shown in figure 1. A black between the groups of blue and red points is an example to differentiate two classes. However, the black line is not the only option to decide the classification boundary. Other than that, if the line moves up a bit and up to just right behind the first blue point meets, the classification is still viable. Alternatively, the line moves down a bit that keeps in front of the first red dot meet, and the classification is also valid. Here we can imagine that the line can be recognized as a correct decision boundary if it stays between two groups of points no matter the line's orientation. Hence, in a theoretical manner, the ways to design the decision boundary can be infinite. By now, we can have a basic concept about the Support Vector Machine, which indeed does the same thing, using a decision boundary or hyperplane to classify two classes.

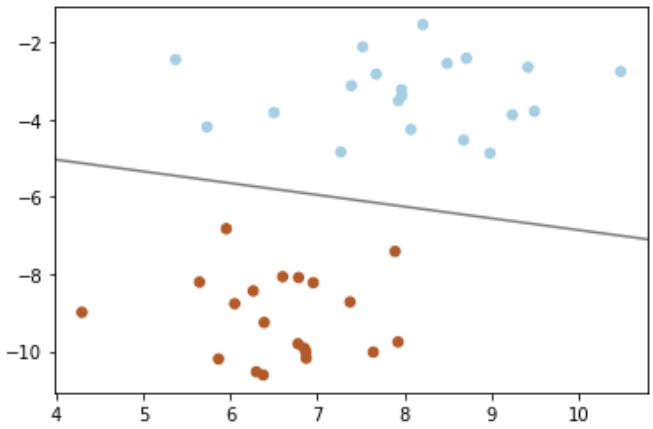


Figure 1. Random scattered points with 2 centers.

SVM is quite similar to the simple linear classification problem mentioned above but reveals more flexibly since the SVM algorithm can tolerate errors by adjusting the margins' width. In contrast, the model discussed above may hard to say if the classification function is still valid when the line touches either one group of point(s). Furthermore, the SVM principle is to create a decision boundary or a hyperplane that can separate the data into two classes. The SVM is also known as a binary classifier. The real-world data can be generally categorized into two types, linearly separable, for example, the data shown in figure 1, and nonlinearly separable, for example, the

data shown in figure 4. For a different type of data, the SVM will have a different approach to handle the data.

$$g(x) = w^T x + b, \text{ where (1)}$$

$$w = \sum_{i=1}^{SV} \alpha_i y_i x_i, and\ b\ is\ the\ bias. \ (2)$$

To define the hyperplanes $g(x)$,

$$w^T x_i + b = -1, \ for\ y_i = -1; (3)$$

$$w^T x_i + b = +1, \ for\ y_i = +1; \ (4)$$

$$w^T x_i + b = 0, \ for\ y_i = 0; (5)$$

Equation (1) – (5) [10].

5.1. Linear-separable case

If we want to train the SVM for the linear-separable case or nonlinear-separable case, a set of training data is required, which should be in input and output pair [6]. The training data is the data that the labels are known. In the SVMs, the training data $X = \{x_1\ x_2\ \ldots\ x_i\}$ will be labeled as either $y_i = \{-1, +1\}$. The purpose of training the SVM is to find the optimal hyperplane. In the linear-separable case, the hyperplane can be illustrated as a simple linear function $f(x) = mx + c$. Through the process of optimization, we can eventually get the optimal hyperplane $g(x)$, equation (1). Before that, we may consider which variables of $g(x)$ are going to be optimized. To optimize $g(x)$, we should consider two conditions, equations (9) and (10).

$$J(w, b, \alpha) = \frac{1}{2} w'w - \sum_{i=1}^{N} \alpha_i y_i w' x_i - b \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i \ (6)$$

$$\frac{\partial J(w,b,\alpha)}{\partial w} = 0 \ \Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i \ (7)$$

$$\frac{\partial J(w,b,\alpha)}{\partial b} = 0 \ \Rightarrow \ \sum_{i=1}^{N} \alpha_i y_i = 0 \ (8)$$

Condition 1: To maximize the margin of the hyperplane d: $\frac{1}{\|w\|}$, where,

$$w = \sum_{i=1}^{SV} \alpha_i y_i x_i \ , sv: number\ of\ support\ vector \ (9)$$

Condition 2: $\sum_{i=1}^{SV} \alpha_i y_i = 0$ (10)

Equation (6) – (10) [6, 10].

For every machine learning algorithm, one of the major tasks is finding the optimal values of weights. Generally, a gradient descent approach will be used to seek the optimal point. For SVM, we have equation (6) to process the gradient descent. In figure 2, the optimal weight is the local minima, where the slope should be 0. Hence, by taking the partial derivative of equation (6) with respect to w and set the derivative equal to 0, the optimal w can be computed. Initially, the weight value is randomly generated and step from the random point to the optimal point through iterations. The step size for each iteration is decided by learning rate—the larger the learning rate, the greater step to proceed. However, if the learning rate is set too large, like 1, there may have a higher chance to miss and step over the optimal point. Hence, in a conventional manner, the learning rate is usually set to 0.01 or lower.

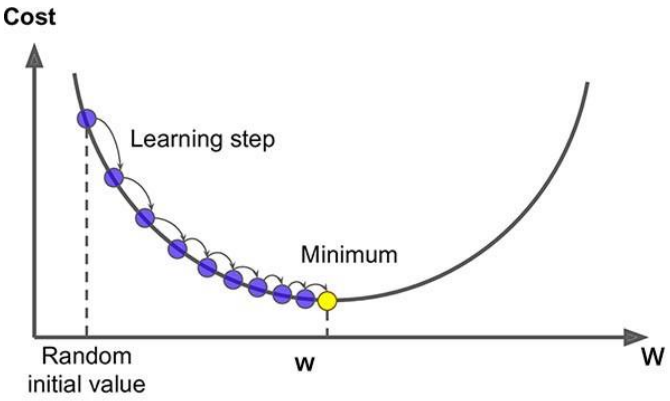


Figure 2. Illustration of gradient descent [8]

$$Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i' x_j \text{ , (11)}$$

$$\frac{\partial Q(\alpha)}{\partial \alpha_i} = 0 \text{ (12)}$$

Equation (11) – (12) [10].

From equation (9), we know that $y_i = \{-1, +1\}$ and x_i are the input sample points while α is the Lagrange multiplier used to compute the weights vector w. We are noted that only α > 0 contributes to optimizing the separating hyperplane. For which stay inside the margin, the α values are less than 0. Hence, by applying equation

(12), the values of $\{\alpha\}_{i=1}^{N}$ can be computed as well. Considering the data are linearly separable, the data points will put in a 2-D plane, the coordinate of each point should be in the form of (x, y), here will be (x_i, x_j) instead. To get $x_i{}'x_j$, let say 3 points are lying on the margins. The support vector consists of only 3 elements $\{ x_1\ x_2\ x_3\}$, where the coordinate of $x_1 = (-1, 1),\ x_2 = (1, 1),\ x_3 = (0, -1)$. Hence, $x_i{}'x_j$ is computed be like:

		x_i		
		x_1	x_2	x_3
x_j	x_1	2	0	-1
	x_2	0	2	-1
	x_3	-1	-1	1

By equation (11), $Q(\alpha) = \sum_{i=1}^{3} \alpha_i - \frac{1}{2}(2\alpha_1^2 + 2\alpha_1\alpha_3 + 2\alpha_2^2 + 2\alpha_2\alpha_3 + \alpha_3^2)$, then taking the partial derivative with respect to $\{\alpha\}_{i=1}^{3}$to 0, equation (12), the solutions of $\{\alpha\}_{i=1}^{3}$ can be found [10]. After computing the α values, the weights vector w can be calculated by equation (7). For the bias b, since it is a scalar value, substituting the w into either equation (3) or (4), b can also be found. The computing processes mentioned here can be regarded as one iteration of calculation. Since all parameters are initialized by some random values, there may take serval iterations to get the optimal point.

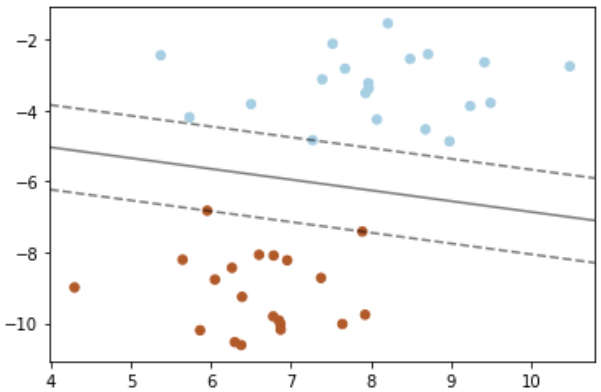


Figure 3. Linear SVM classifier.

5.2. Nonlinear-separable case

The nonlinear-separable data may require a soft margin to separate. For a supplement, the hyperplane used in linear-separable data is also known as hard margins. The soft margin or the nonlinear property shows that the hyperplane can be a curve in a low dimension space. However, if the nonlinear-separable data are being put into a high dimensional space, a flat hyperplane can be used to separable the data. Since in a very high dimensional space, like $\mathbb{R}^{n>no.\ of\ sample\ data}$, the data points can be easily separated, as shown in figure 4. Hence, a simple flat hyperplane can do the job of separation [9].

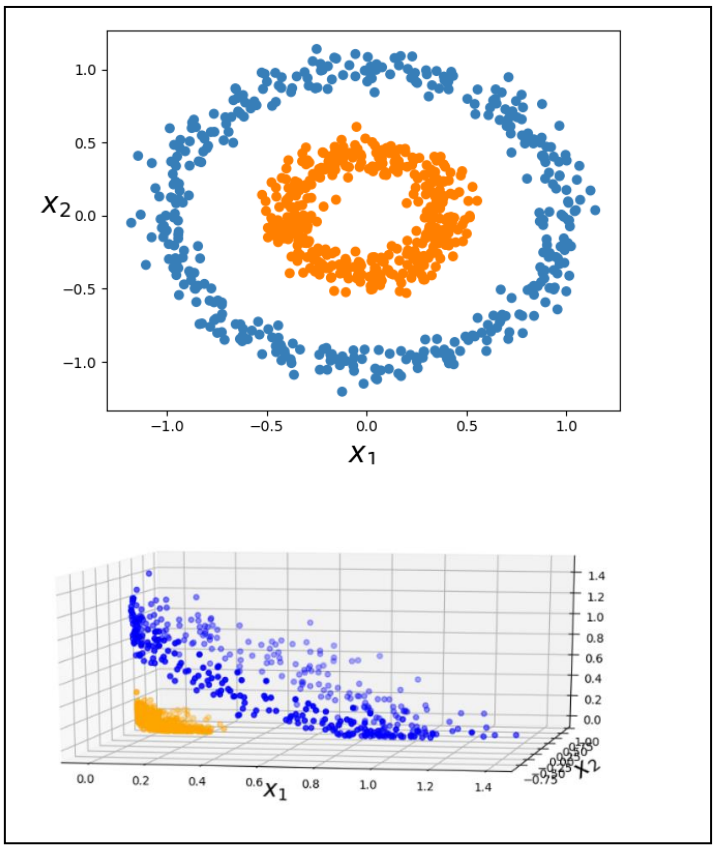


Figure 4. Illustration of nonlinear-separable data [9].

6. Application

Face detection with the SVM classifier will be introduced. Face detection, also known as facial detection, is a machine learning-based algorithm commonly used to find and identify human faces in a digital image [1]. It is a crucial component that associates with different applications, including surveillance, biometrics, entertainment, and personal safety [1].

7. Methodology

7.1. Feature extraction

In this project, the Histogram of Oriented Gradients (HOG) features will be extracted as the input feature vector. The HOG features are used since the extraction of HOG features is easy to implement and has a quite decent success rate in object detection. The HOG features are based on the orientation of the pixel values of Gx and Gy. By summing the gradient magnitudes concerning the relative orientations. The orientation with the highest magnitude represents the edge that more likely exists in that orientation. The closer to the edge, the greater the response of the orientation. Figure 5 illustrates the relationship between the edge and the orientation. From the figure, the orientation of 180-degree gets the sharpest response, which means the orientation of 180-degree is the closest direction to the edge. Base on this principle, the HOG can locate the object's edges and summarize the gradients and magnitudes as the input features vector.

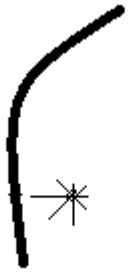


Figure 5. Illustration of HOG features of one block-cell.

The HOG is a feature descriptor, which has been widely used in computer vision tasks and object detection [3]. By applying the steps mentioned below, the HOG features can be extracted from an image. We should note that the dimension of HOG features depends on the block size and number of orientations, besides the image dimension. For example, in figure 7, the block size is 4x4, and each block-cell will take a 9-orientation histogram, for each block-cell will include 3x3 pixels. Thus, one 4x4 block can extract 4x4x9 features.

From the beginning, an image will be first divided into n overlapped or non-overlapped blocks. Each block is formed by $m \times m$ cells. Each block cell may include $l \times l$ pixels to compute the $h\ orientations$ HOG of a block cell. Lastly, a $n \times m \times m \times h$ dimensions feature vector will be produced for the image. The gradient magnitude of each pixel can be calculated by $G = \sqrt{G_x^2 + G_y^2}$, while the orientation of the gradient of each pixel can be retrieved by $\tan^{-1}\frac{Gx}{Gy}$, where $G_x = I \otimes S_x$ and $G_y = I \otimes S_y$, and S_x and S_y are the Sobel operators. The gray-scale image convolutes with the Sobel operators to reserve the face's edges [10].

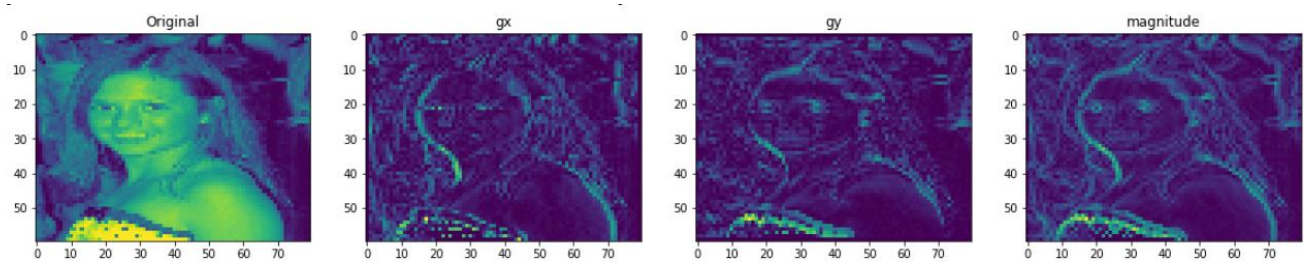


Figure 6. Images with the Sobel filter, original label is the gray-scaled image, gx image is the result image after applying Sx and gy image is the result image after applying Sy. The magnitude label image is magnitude image of Gx and Gy.

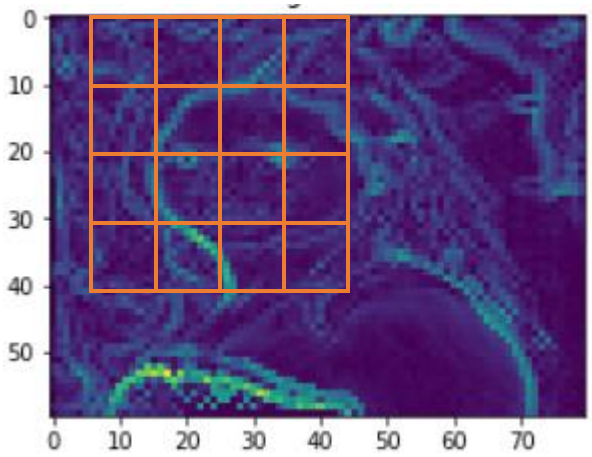
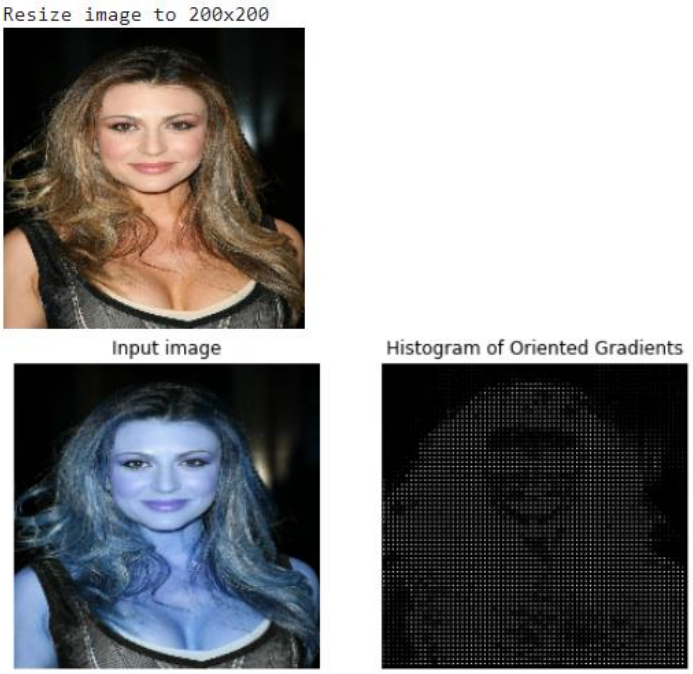
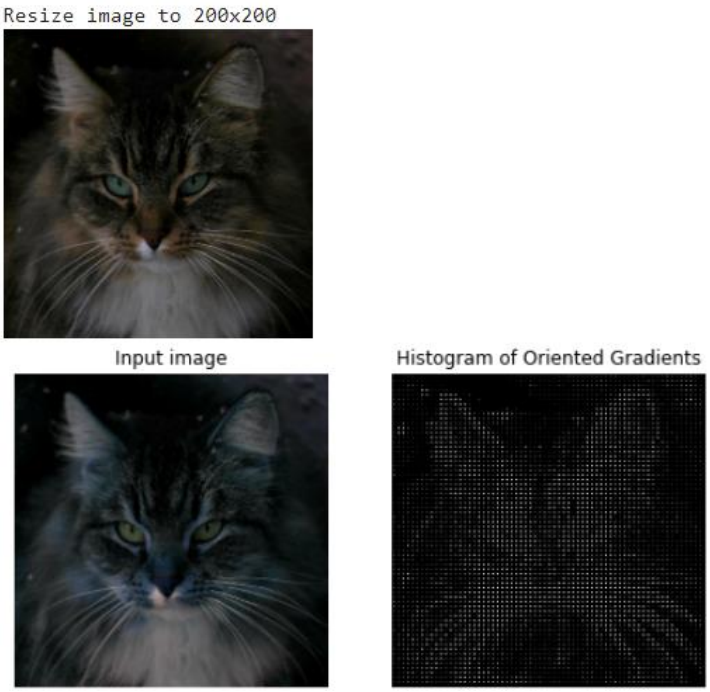


Figure 7. A 4x4 block is scanning through the image to extract the HOG features.





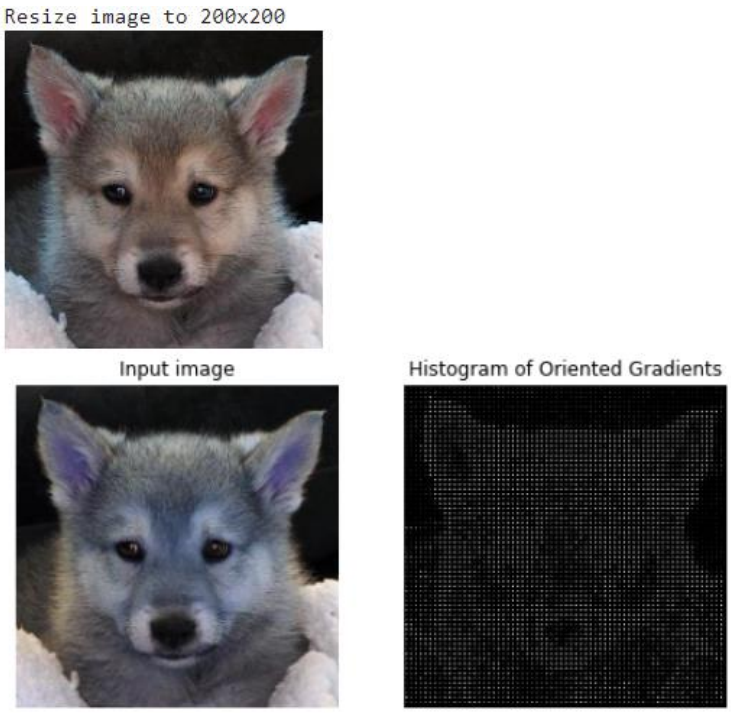


Figure 8. using 4x4 block, 3x3 pixels per block-cell, and 9-orientation histogram to extract the HOG feature.

Figure 8 demonstrates the HOG featured image. The features usually are flattened into a 1-D vector for use as the input vector. We use the sklearn toolkit to compose the original image dimensions with the HOG features to see the differences between before and after extracting features. The left of the figure is an original image, and the right-hand side is a HOG composed image. From the formed image, we can see that the edges of the images are emphasized substantially.

7.2 Model training approach

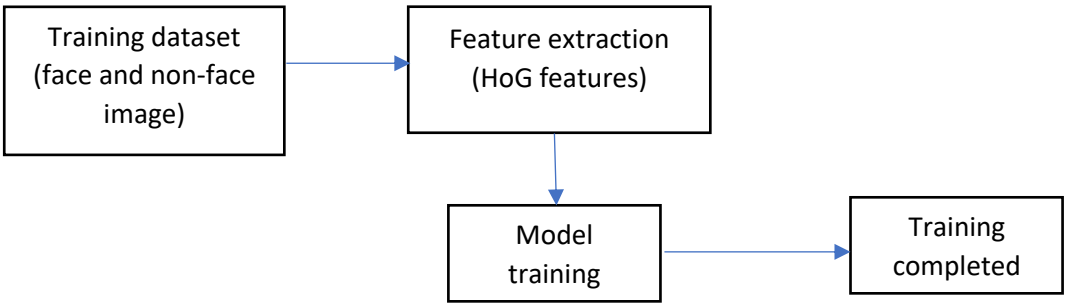


Figure 9. Model training workflow.

Figure 9 has demonstrated a general model training approach. First, we may collect enough images to train the model to converge at a certain point by the mean of gradient descent. Usually, model training requires a thousand iterations to make it converge.

7.3. Detection with sliding window and image scaling

In order to identify and locate the human faces in a digital photo, a fixed size window will apply to the picture. The window acts like a scanner that scans through the image. The step size parameter can determine the scanning speed of the window. For example, an image with 400x600 (height x width) dimensions uses a 30x30 window to detect. The number of samples for the first-row iteration of the image is: $\frac{width}{step\ size} = \frac{600}{2} = 300\ samples$ will be fed into the model, and the samples will be predicted sequentially. If the step size increases to 5, the number of one-row iteration will reduce to 120. Of course, a tradeoff between scanning speed and precision occurs since if the step size is set too high, the scanning window will skip some substantial pixels that the window cannot truly predict the potential faces. Hence, the faster the detection speed, the lower the precision of the detection. It is known as one of the limitations of this approach to detect the faces.

Since the sliding window is fixed in size, another problem occurs when the face in the image is larger than the window size that the window can only capture part of the

face, which can cause a false negative as a result. Hence, a downscaling process of the image is required right after scanning through with the previous image scale. In the application, the downscale factor will be 0.9. The original image scale is unchanged except for the last resize process (each test image will be resized to width = 600 pixels before undergoing the scanning process). It means that the image will be kept downscaling until the final image is equal to or smaller than the sliding window size.

7.4. Non-maximum Suppression (NMS)

The same face detection can be triggered multiple times if the sliding window is applied to the image. As the step size should be as small as possible to have better performance, the true face can be detected frequently. Also, the detection is dependent on detection confidence. Once the frame confidence is higher or equal to the predefined threshold, which is normally set as 0.5, the model will regard it as a positive detection. This is the reason why causing such many redundant detections in figure 10.

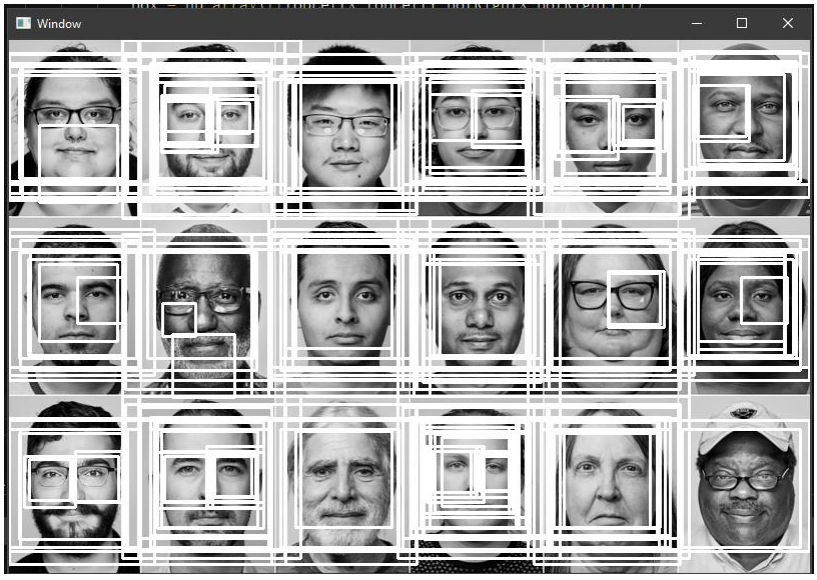


Figure 10. Detection result without using NMS.

To eliminate the redundant detection windows figure 13, a technique that filters the redundant proposals based on certain criteria is called NMS. The collection of proposals in terms of the input parameter of the NMS function will be in the form of

$[[x1, y1, x2, y2, \text{score} \geq \text{the predefined threshold}], \ldots]$, where (x1, y1) and (x2, y2) represent the top-left coordinates and the bottom-right coordinates, respectively. The proposals and the relevant confidence scores are stored in the same list. As mentioned in the previous part, the acceptance of the proposal is based on the predefined threshold, which is usually 0.5.

To apply NMS in figure 12, we may conduct the following procedures:

1. The proposal in $list\ B$ will first sort in descending order according to the confidence score. The proposal with the highest confidence score is popped out and appended to another empty $list\ B_{nms}$ which will integrate the output proposals at the end of the process.

2. Then, the Intersection Over Union (IOU) should be considered so that the choice can be made to give up any of the proposals. Also, the choice is dependent based on the IOU threshold. The threshold represents the degree of tolerance of the proposal overlapping. The larger the threshold, the higher degree of tolerance in overlapping. If the IOU is larger than the threshold, the proposal is removed from $list\ B$. To calculate the IOU, we may refer to figure 11, using the intersection area over the union area of two overlapped frames.

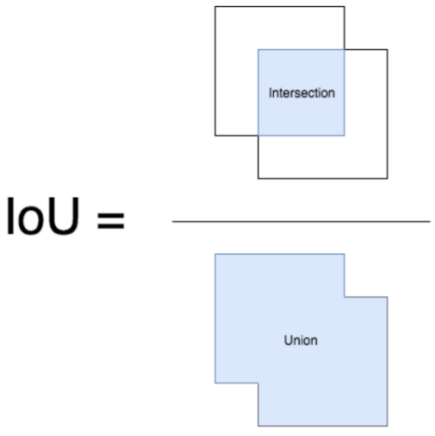


Figure 11. Intersection over union calculation.

3. Revert and start from step 1 again.
4. The processes repeat until no proposal left in $list\ B$.

```
1:  procedure NMS(B,c)
2:      B_nms ← ∅          Initialize empty set
3:      for b_i ∈ B do     => Iterate over all the boxes
4:          discard ← False    Take boolean variable and set it as false. This variable indicates whether b(i) should be kept or discarded
5:          for b_j ∈ B do     Start another loop to compare with b(i)
6:              if same(b_i, b_j) > λ_nms then    If both boxes having same IOU
7:                  if score(c, b_j) > score(c, b_i) then
8:                      discard ← True    Compare the scores. If score of b(i) is less than that of b(j), b(i) should be discarded, so set the flag to True.
9:          if not discard then    Once b(i) is compared with all other boxes and still the discarded flag is False, then b(i) should be considered. So add it to the final list.
10:             B_nms ← B_nms ∪ b_i
11:     return B_nms    Do the same procedure for remaining boxes and return the final list
```

Figure 12 Pseudo code of the NMS [12].

Figure 13. Detection result by using the NMS with 0.3 overlap threshold.

8. Experiment and results

In this section, an experiment will be conducted to evaluate the performance of the application. The images used in the testing are majority from the Internet. The searching keywords are: {person, group of people, front faces}.

8.1. Training dataset

The positive training dataset is from Caltech Vision Group (figure 14) [13], and only portrait images are used for training. The dataset has a total of 6714 face images that can be used for training. For the negative training samples, the samples are the fragments of the images with no face pattern. In practice, the negative sample data can be generated by random arrays. Both fragments of the images with no faces and random arrays images are viable to train the face detection model since they are valid to be counted as negative data as long as there is no face. The dimensions of the training samples should be the same so that the consistency of feature dimensions can be maintained.

Figure 14. Caltech web face dataset [13].

Workstation specifications:

CPU: AMD Ryzen 7 3700X

RAM: 16GB DDR4 Memory

Operation Environment: Window 10 Pro 64-bit

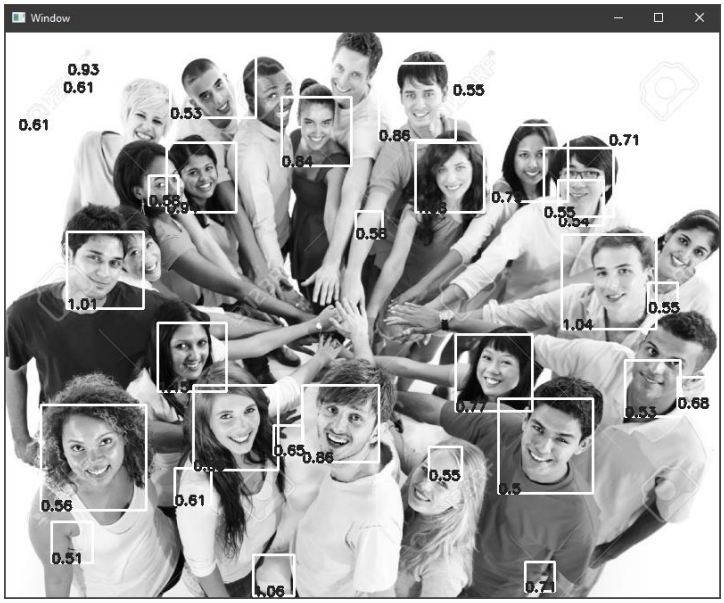
Program IDE: PyCharm

8.2. Results

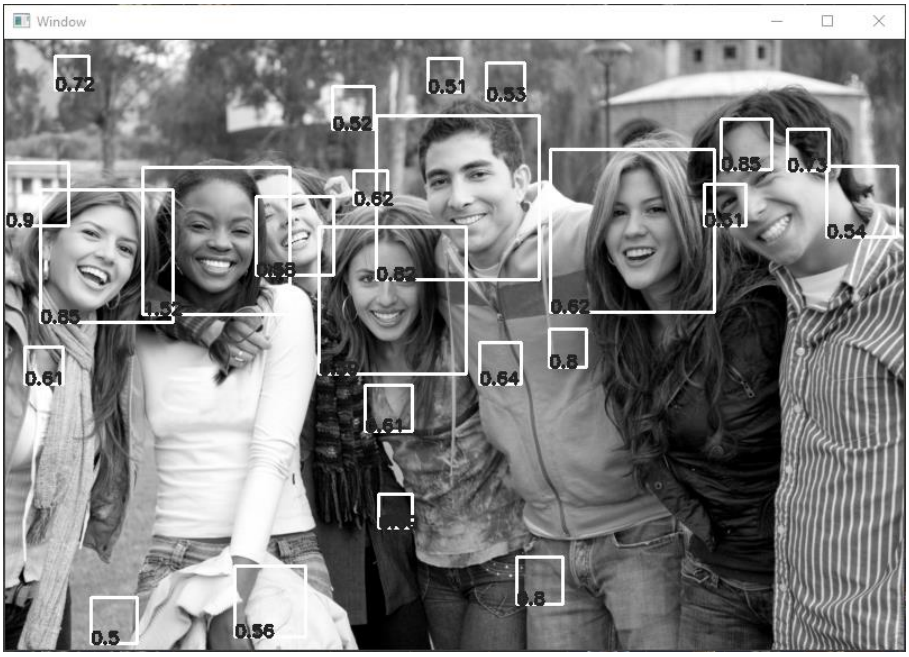
Detection confidence is attached along with the detection window at the bottom left.

1. Detection threshold = 0.5, step size = 2, NMS threshold = 0.5. Number of Negative samples: 100000, number of positive samples: 6000, feature dimensions = 1600.

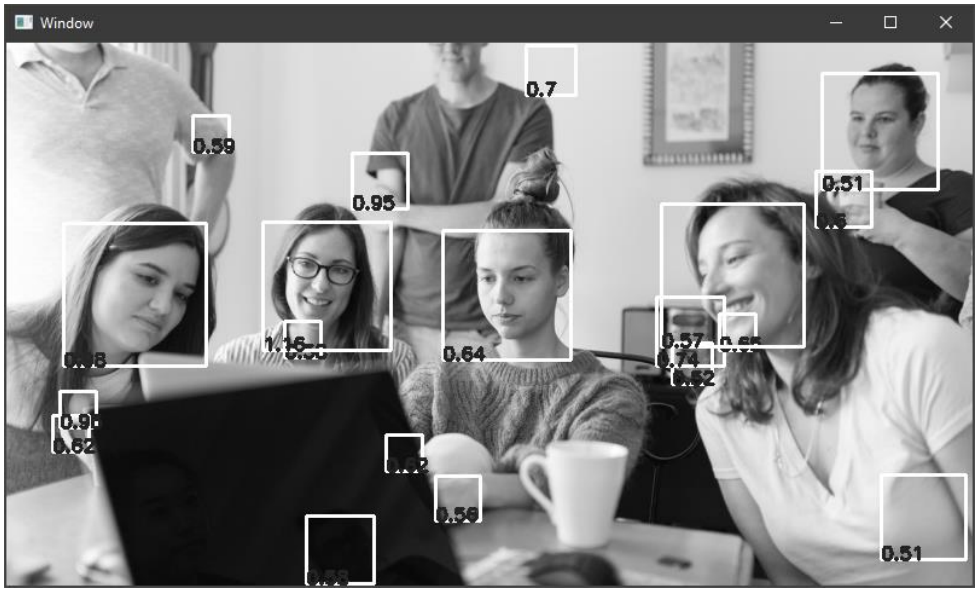
Test image 1.1.



Test image 1.2.

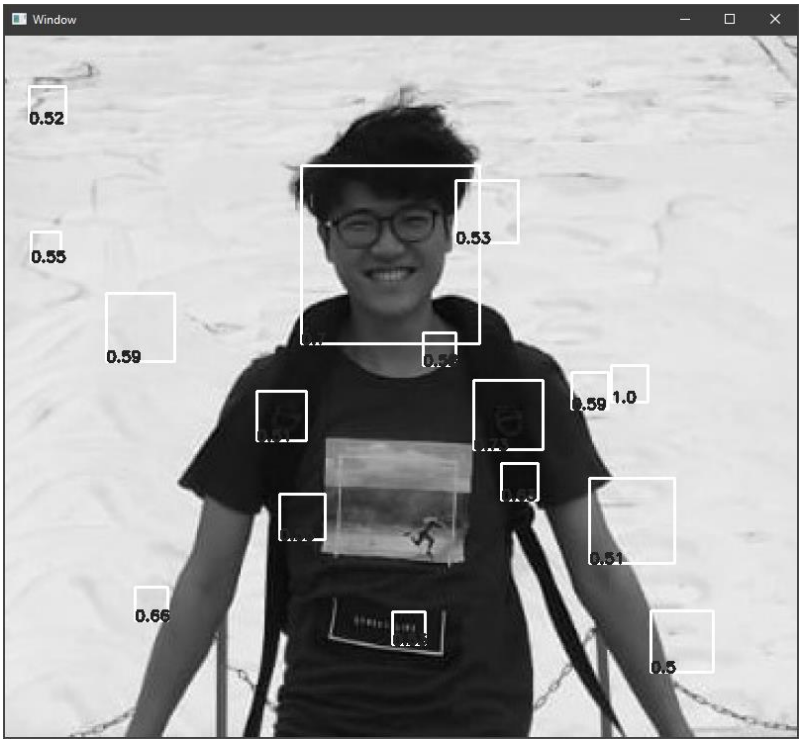


Test image 1.3.

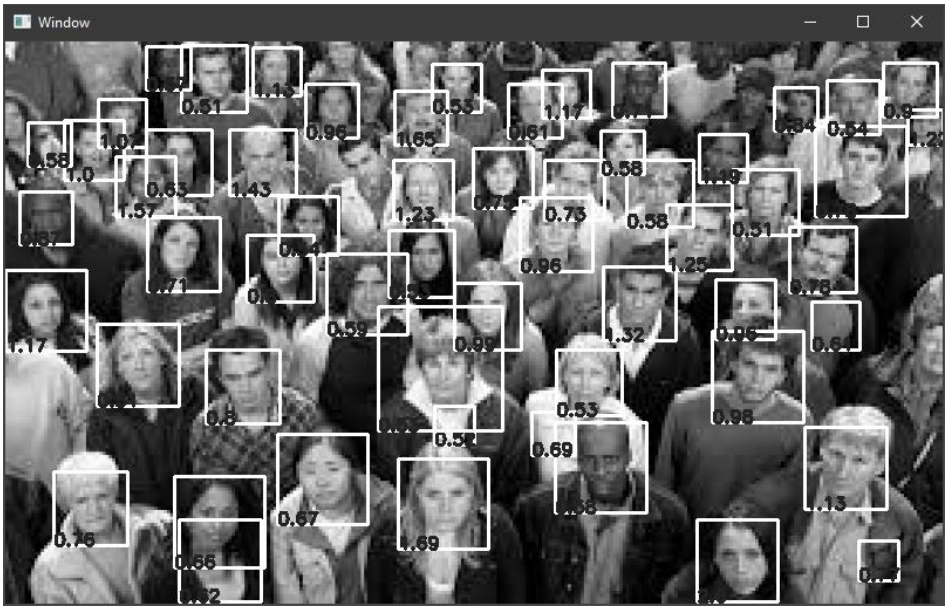


Test image 1.4.

Test image 1.5.



Test image 1.6.



Test image 1.7.

	Image dimension	Time used (second)	Total number of detections	Number of ground truth faces	Number of detected faces	Detected face to total detections	Detection accuracy
1.	533x800	38.9	19	18	18	0.95	100%
2.	633x800	46.94	32	23	15	0.47	60.87%
3.	543x800	38.68	24	7	6	0.25	71.43%

4.	449x800	32.28	19	5	5	0.26	100%
5.	504x800	37.24	20	9	9	0.45	100%
6.	709x800	54.89	16	1	1	0.06	100%
7.	480x800	35.3	57	56	52	0.91	92.86%

Table 1.

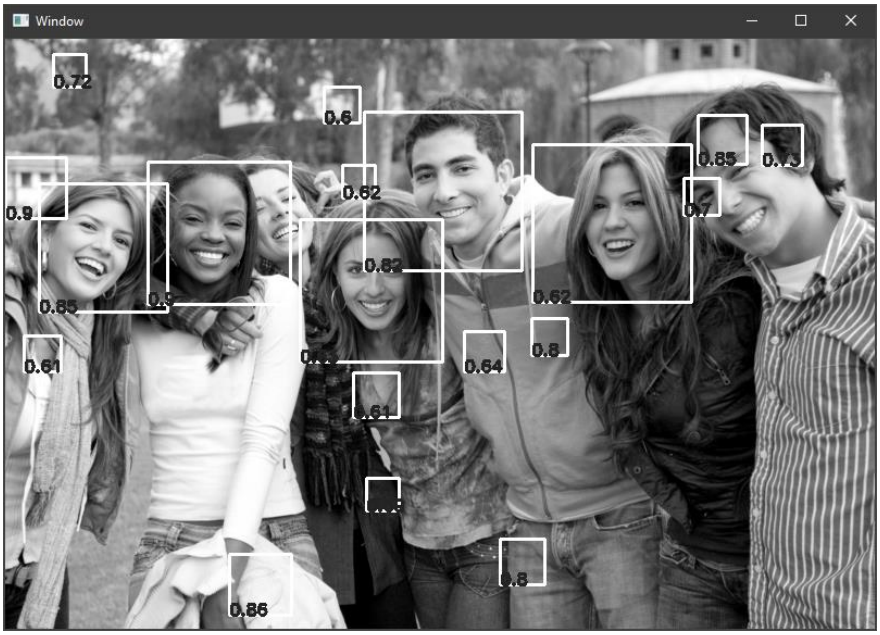
	Total number of detections	Number of false positives	Percentage of false positive detections
1.	19	1	5.26%
2.	32	15	53.13%
3.	24	18	75%
4.	19	13	72%
5.	20	9	50%
6.	16	10	90.9%
7.	57	4	7%

Table 1.1.

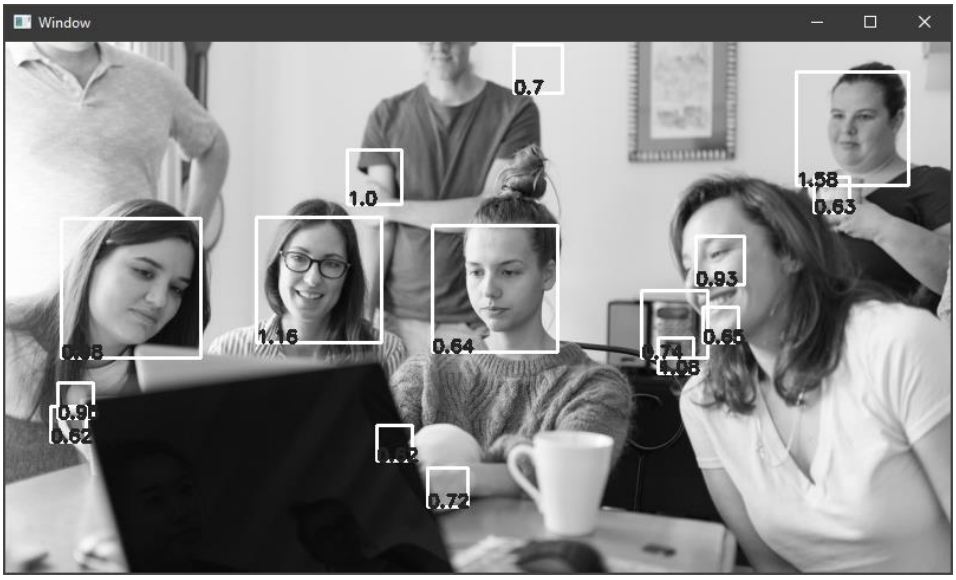
2. Detection threshold = 0.6, step size = 2, NMS threshold = 0.6. Number of Negative samples: 100000, number of positive samples: 6000, feature dimensions = 1600.

Test image 2.1.

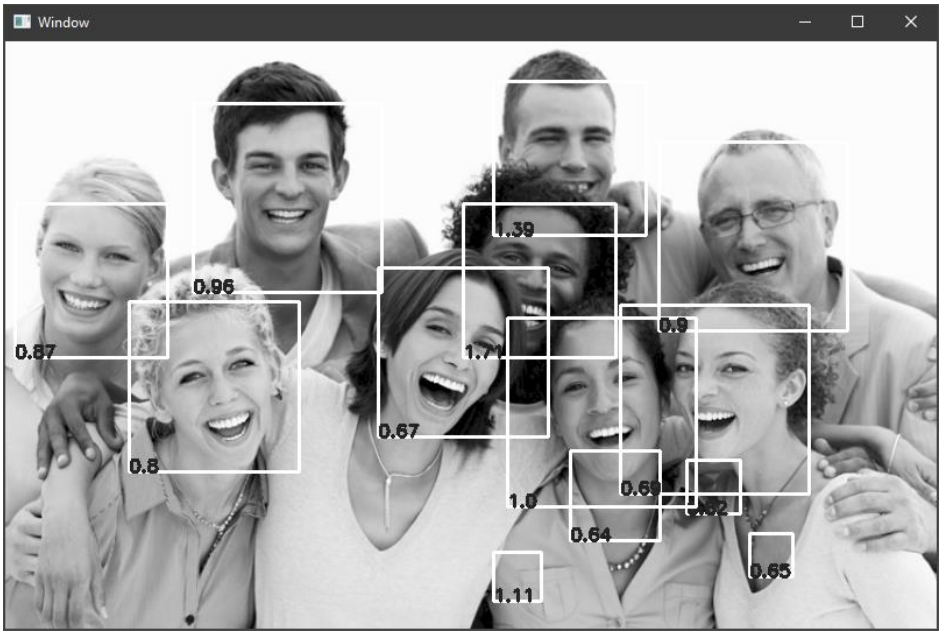
Test image 2.2.



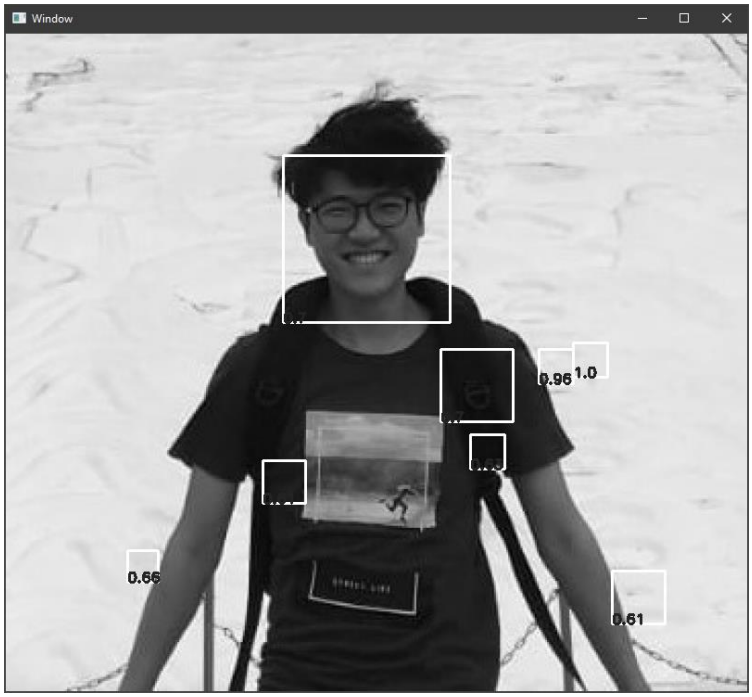
Test image 2.3.



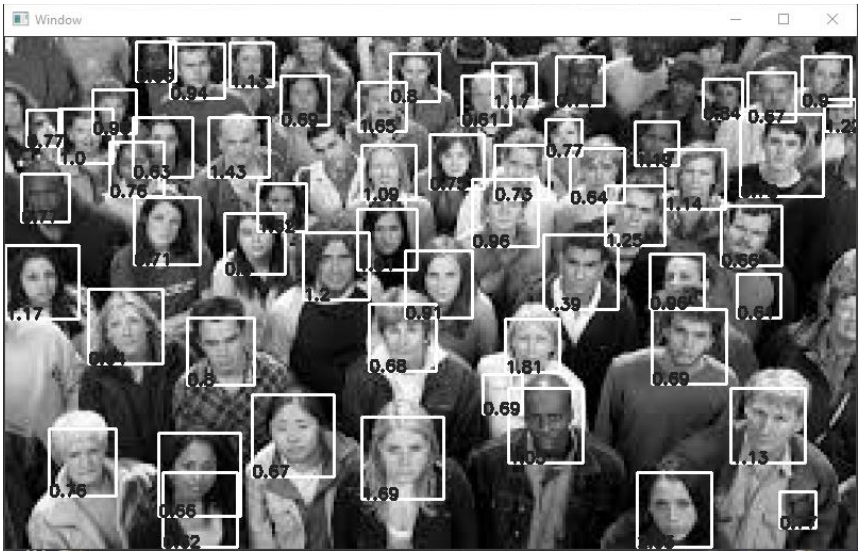
Test image 2.4.



Test image 2.5.



Test image 2.6.



Test image 2.7.

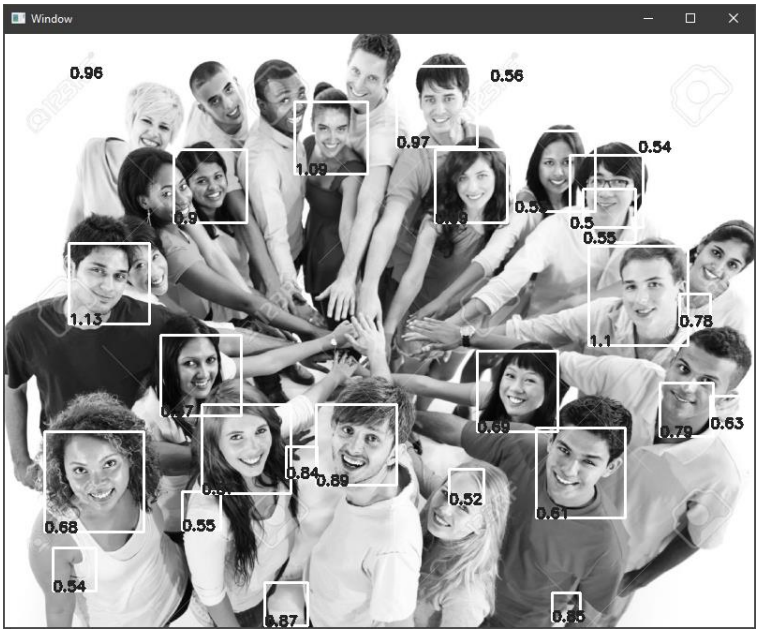
	Image dimension	Time used (second)	Total number of detections	Number of ground truth faces	Number of detected faces	Detected face to total detections	Detection accuracy
1.	533x800	37.75	19	18	18	0.95	100%
2.	633x800	47.23	27	23	14	0.52	60.87%

3.	543x800	39.96	19	7	5	0.26	71.43%
4.	449x800	32.21	15	5	4	0.27	80%
5.	504x800	33.76	13	9	9	0.5	100%
6.	709x800	49.51	8	1	1	0.125	100%
7.	480x800	31.71	56	56	52	0.93	92.86%

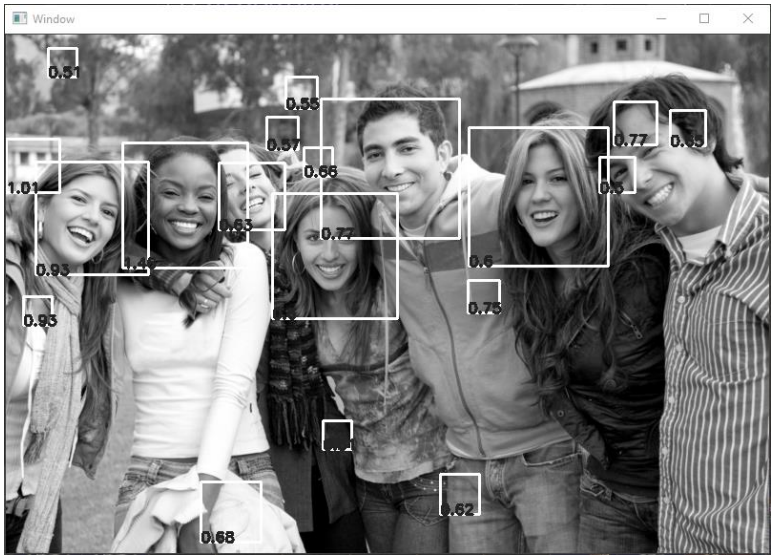
Table 2.

3. Detection threshold = 0.5, step size = 2, NMS threshold = 0.5. Number of Negative samples: 140000, number of positive samples: 6700, feature dimensions = 1600.

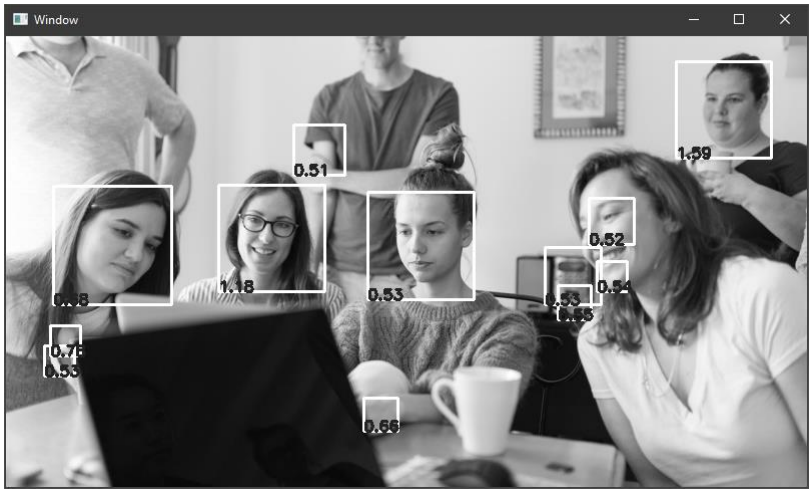
Test image 3.1.



Test image 3.2.

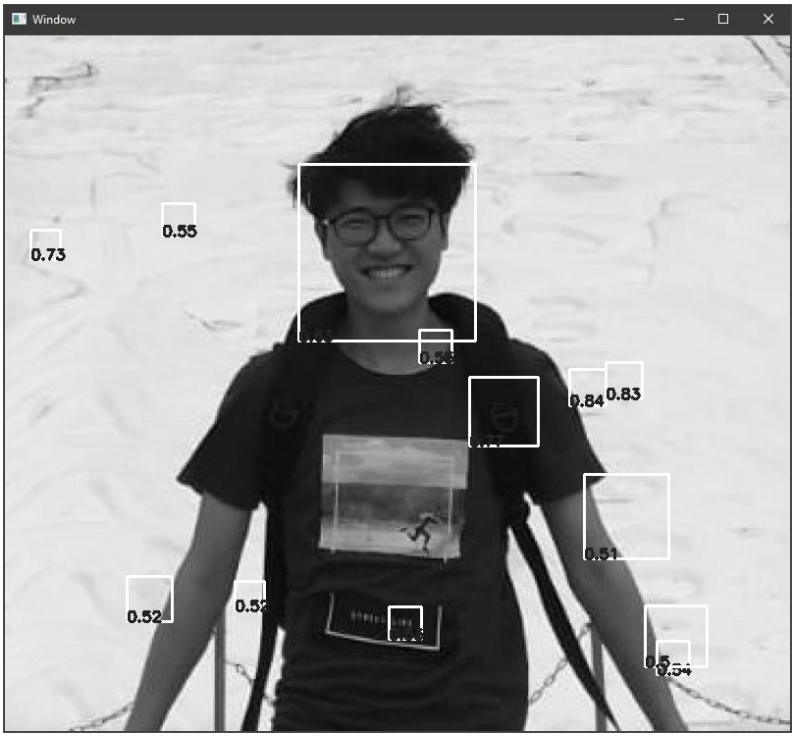


Test image 3.3.

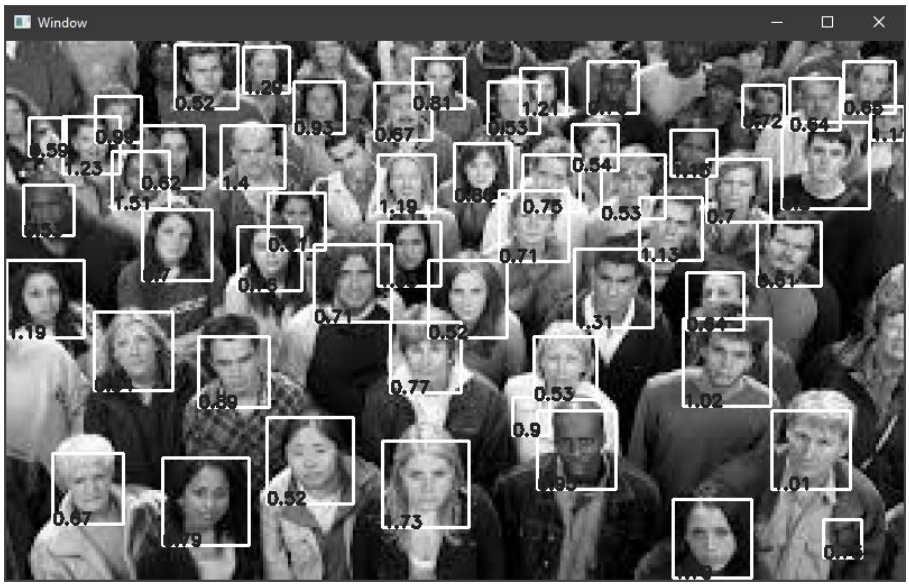


Test image 3.4.

Test image 3.5.



Test image 3.6.



Test image 3.7.

	Image dimension	Time used (second)	Total number of detections	Number of ground truth faces	Number of detected faces	Detected face to total detections	Detection accuracy
1.	533x800	37.78	19	18	18	0.95	100%

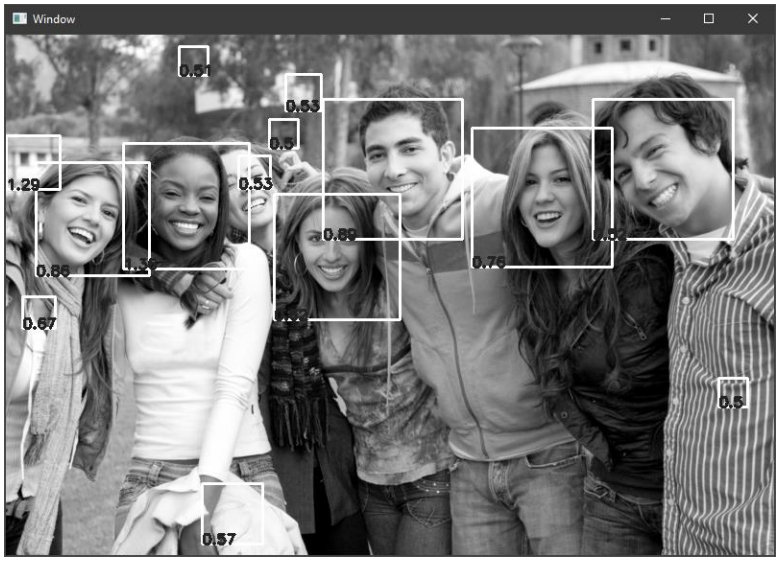
2.	633x800	45.48	27	23	14	0.52	60.87%
3.	543x800	37.64	19	7	6	0.32	85.71%
4.	449x800	30.34	12	5	4	0.33	80%
5.	504x800	34.2	19	9	9	0.47	100%
6.	709x800	52.18	13	1	1	0.08	100%
7.	480x800	33.41	53	56	51	096	91.1%

Table 3.

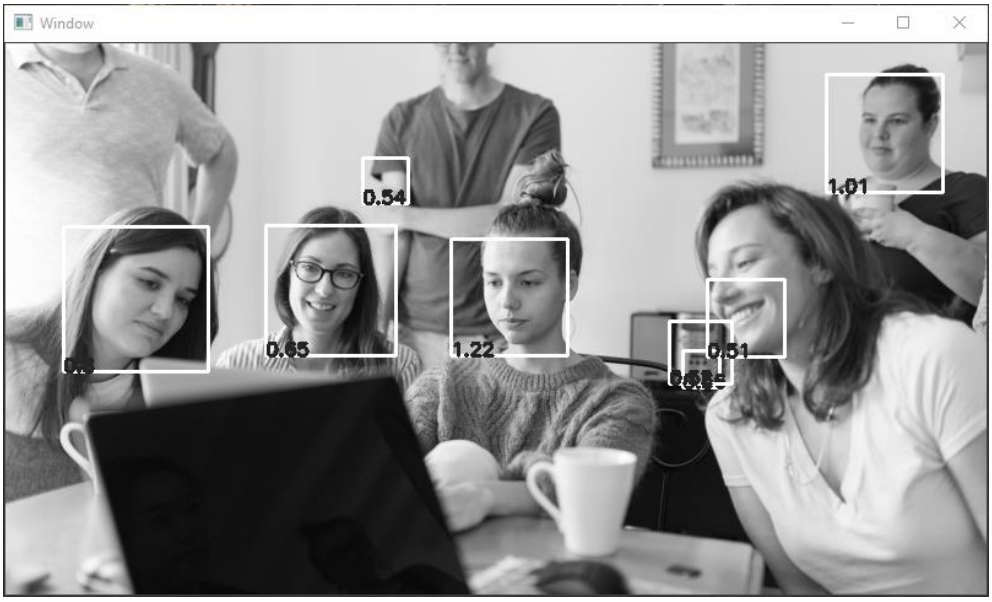
4. Detection threshold = 0.5, step size = 2, NMS threshold = 0.5. Number of Negative samples: 140000, number of positive samples: 6700, feature dimensions = 10000.

Test image 4.1.

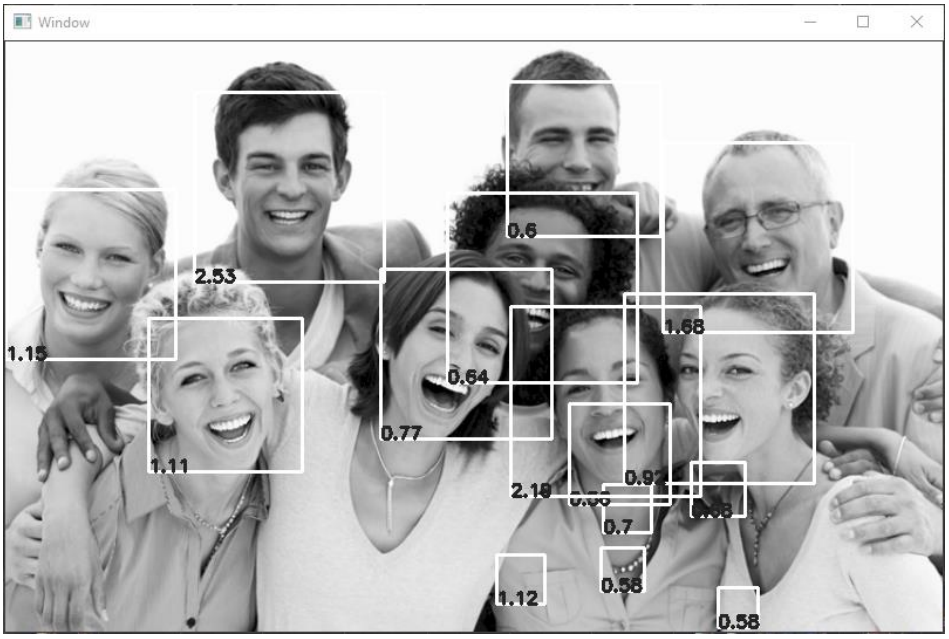
Test image 4.2.



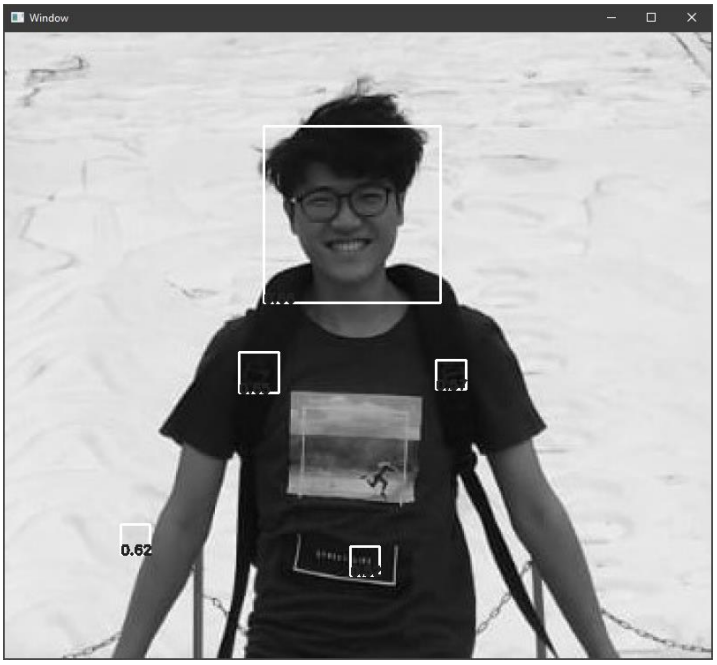
Test image 4.3.



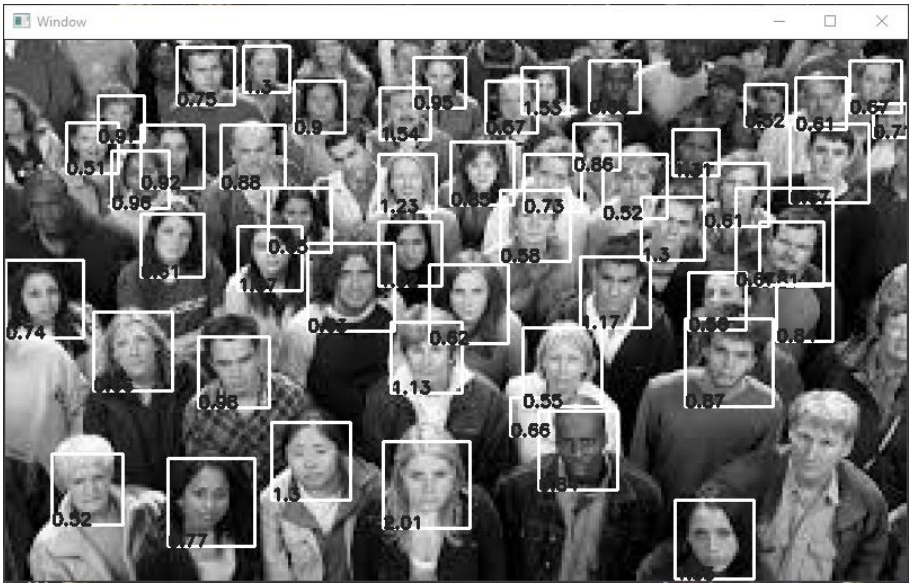
Test image 4.4.



Test image 4.5.



Test image 4.6.



Test image 4.7.

	Image dimension	Time used (second)	Total number of detections	Number of ground truth faces	Number of detected faces	Detected face to total detections	Detection accuracy
1.	533x800	46.65	20	18	18	0.9	100%
2.	633x800	61.48	22	23	14	0.64	60.87%
3.	543x800	52.14	14	7	6	0.43	85.71%
4.	449x800	56.13	8	5	4	0.5	80%
5.	504x800	74.15	15	9	9	0.6	100%
6.	709x800	82.16	5	1	1	0.2	100%
7.	480x800	40.12	51	56	48	0.94	85.71%

Table 4.

In this experiment, the detection accuracies of both trials are more or less the same. The average accuracy of these seven images is around 85%. Some faces are hard to detect. For example, the faces are not directly facing the camera, or the faces tilt to the side. The best situations for the application can detect correctly are, the human faces in the image look straight to the camera. If the faces have no turning and no tilting, the model can produce accurate results (test image 1). Besides, the face detector produces many false-positive results, as mentioned in the methodology part.

The false-positive problem can be slightly improved by feeding more negative samples in the training process. The tables1-2 and table 3-4 have demonstrated that increases the number of negative samples from 100,000 to 140,000 can reduce some false positive detections. Also, increasing the detection threshold is the most effective way to avoid false-positive detections. Table 1 and Table 2 show that if the threshold increases, there are fewer detection windows, which means increasing the threshold does prevent false positive detections. But the face detection accuracy may also be affected since the detector may discard some true-positive detections if their detection confidences are not high enough. To further eliminate the false positive detections, increasing the feature spaces can be the other way to achieve. Tables 3 and 4 reveal that the false positive detections can be further eliminated by expanding the feature dimensions from 1600 to 10000. Higher feature dimensions mean more face features will be retained as the distinctions between the true face and non-face detection.

9. Conclusion

I have introduced the SVM in the previous stage of the project and how it can be applied in image classification (classify cat, dog, and human). The robustness of the SVM as the general classifier can apply to different applications apart from pattern recognition due to the kernel property. In phase II, to further review the potential of the SVM to recognize the more complex pattern, I develop the face detector and evaluate its advantages and disadvantages. For the advantages, the performance of the SVM is surprisingly decent with small among of training data where I discovered in the image classification experiment, 100 images for each class without using any feature descriptor, and it still can maintain a decent accuracy which can reach about 90% accuracy. Furthermore, the training speed is fast, which helps use the SVM classifier to evaluate the feasibility of a machine learning project, even though it may not be the best choice of classifier for the project. Moreover, the model file size is small, even using hundred-thousand data in training, since the model file can keep the weights of the support vectors.

For the disadvantages, the SVM is not suitable for training with a large dataset since the weights may be hard to converge with limited iterations. Also, the tolerance of noises is not good. For example, if the training data for face detection contains many background pixels, the classifier's performance will decrease substantially.

The overall results of the face detector satisfy my expectations, although limited by the sliding window approach, which arouses quite a lot of false-positive detections. The HOG features are great for reducing the dimensionality of the data, and the feature patterns can still be maintained reasonably well. Based on the HOG features, most of the faces can be detected successfully.

Perhaps, the application can be further improved and developed through deep learning algorithms, like convolution neural networks (CNN). Due to the spatial invariance, the CNN-SVM model with HOG descriptor can be expected to handle the situation of face turning and tilting.

10. References

[1] M. Rouse, “What is Face Detection and How Does It Work?,” SearchEnterpriseAI, 03-Feb-2020. [Online]. Available: https://searchenterpriseai.techtarget.com/definition/face-detection. [Accessed: 22-Oct-2020].

[2] P. J. Philips, “Support Vector Machines Applied to Face recognition,” National Institute of Standards and Technology, Nov. 1998.

[3] A. Singh , “Feature Descriptor: Hog Descriptor Tutorial,” Analytics Vidhya, 04-Sep-2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/. [Accessed: 22-Oct-2020].

[4] Laurent H. “AlSegment.com – Matting Human Datasets”, Kaggle, 07-Jun-2019. [Online]. Available: https://www.kaggle.com/laurentmih/aisegmentcom-matting-human-datasets. [Accessd: 22-Oct-2020].

[5] Larxel. “Animal Faces”, Kaggle, 22-May-2020. [Online]. Available: https://www.kaggle.com/andrewmvd/animal-faces. [Accessd: 22-Oct-2020].

[6] R. Berwick, “An Idiot’s guide to Support vector machines (SVMs),” in *MIT*, 01-Jan-2021.

[7] B. BAŞA*, “Implementation of hog edge detection algorithm onfpga’s,” *Procedia - Social and Behavioral Sciences*, vol. 174, no. 2015, pp. 1567–1575, 2015.

[8] A. Sauer, “Quick Guide to Gradient Descent and It’s Variants,” *Social Networks for Programmers and Developers*, Aug-2020. [Online]. Available: https://morioh.com/p/15c995420be6. [Accessed: 01-Jan-2021].

[9] D. Wilimitis, “The Kernel Trick in Support Vector Classification,” *Medium*, 12-Dec-2018. [Online]. Available: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f. [Accessed: 01-Jan-2021].

[10] K. Lam and Z. Chi, “Computer Vision and Pattern Recognition,” in *EIE4100*, 01-Jan-2021.

[11] K. Rawat, “Tutorial: Linearly separable data,” *CommonLounge*, 01-Jan-2021. [Online]. Available: https://www.commonlounge.com/discussion/6caf49570d9c4d0789afbc544b32cdbf. [Accessed: 01-Jan-2021].

[12] S. K, “Non-maximum Suppression (NMS),” *Medium*, 23-Feb-2021. [Online]. Available: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c. [Accessed: 21-Apr-2021].

[13] M. Fink, “Caltech 10, 000 Web Faces,” *Caltech 10,000 Web Faces*. [Online]. Available: http://www.vision.caltech.edu/Image_Datasets/Caltech_10K_WebFaces/. [Accessed: 21-Apr-2021].