Real Time Human Detection and Counting

A *Machine Learning Minor Project Report*

Submitted for partial fulfilment of the award of

Bachelor of Technology

Submitted By:

Indrajeet Kumar Ranjan (202086)

Parth Tyagi (202100)

Under the guidance of

Dr. Sushil Kumar

Assistant Professor

Department of Computer Science & Engineering

Central University of Haryana

Department of Computer Science & Engineering

School of Engineering & Technology

Central University of Haryana, Mahendergarh

Haryana - 123031, India

December,2023

Declaration

We hereby declare that the work being presented in this project entitled," **Real Time Human Detection & Counting** " by us, Indrajeet kumar ranjan(202086), Parth Tyagi(202100) in partial fulfilment of the requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the **Department of Computer Science and Engineering at** Central university of Haryana is an authentic record of our own work carried out under the supervision of **Dr. Sushil Kumar**. The matter presented in the report has not been submitted in any other University/Institute for the award of any degree.

Indrajeet Kumar Ranjan **Parth Tyagi**

This is to certify that above statement made by the candidate is correct to the best of our knowledge

Dr. Sushil kumar
(Asst. Professor)
Countersigned By:

The B.Tech Project Viva-voce examination of **Indrajeet Kumar Ranjan** and **Parth Tyagi** has been held on and is accepted.

Dr. Rakesh Kumar
(Head of Department) **(Examiner)**

Acknowledgement

First of all we would like to take this opportunity to thanks our supervisor **Dr. Sushil Kumar** without whose efforts this project would not have been possible. We are grateful to him for guiding us towards the project wherever possible. We are most grateful to Department of Computer Science and Engineering, Central University of Haryana for providing us this wonderful opportunity to complete our 4th year project.

And last but the biggest of all, We want to thank to each of the group members, for always helping keeping a continuous check that project never wandered off the track from our goal.

Indrajeet Kumar Ranjan (202086)
Parth Tyagi (202100)

Abstract

This project investigates and reports benchmarks for detecting and enumerating humans through real time images, videos, and camera.
This is very useful in various image processing and performing computer vision tasks.
This schemes have been implemented in Python programming language, and using various tech-stacks like OpenCV, TensorFlow, etc.

Keywords : Computer Vision, Human Detection, Enumeration

Table of Content

Chapter 1

1. Introduction

This chapter resembles the brief introduction about the most widely used field of study “Computer Vision”. Here talked about the various aspects and uses of computer vision, basic meaning and keywords like detection, enumeration, and discussed the roadmap to the report.

2. Computer Vision

Computer vision is a multidisciplinary field that enables machines to interpret and make decisions based on visual data from the world. It involves the development of algorithms and models that allow computers to gain high-level understanding from images or videos. The primary goal of computer vision is to enable machines to perform tasks that typically require human vision.

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. It is most widely used field of artificial intelligence.

Figure1.1

Key tasks and concepts in computer vision include:

2.1.Image Processing:

Techniques for manipulating and enhancing digital images, such as filtering, edge detection, and image segmentation.

2.2.Object Detection:

Identifying and locating objects within images or video streams. Object detection algorithms often use bounding boxes to outline the detected objects.

2.3.Image Classification:

Assigning predefined labels to images based on their content. This is a fundamental task in computer vision and often involves training machine learning models.

2.4.Image Recognition:

Generalizing image classification to recognize and categorize objects or scenes in images.

2.5.Facial Recognition:

Identifying and verifying individuals based on their facial features.

2.6.Gesture Recognition:

Interpreting gestures made by humans, typically using cameras to capture and analyze hand movements.

2.7.Semantic Segmentation:

Assigning semantic labels to each pixel in an image, allowing for a more detailed understanding of the scene.

2.8.3D Computer Vision:

Extracting three-dimensional information from two-dimensional images or multiple images, often for applications like 3D reconstruction.

2.9.Feature Detection and Matching:

Identifying and matching key features or points in images, which is crucial for tasks like image stitching or object tracking.

2.10. Deep Learning:

Leveraging neural networks, especially convolutional neural networks (CNNs), to automatically learn hierarchical features from data, improving performance on various computer vision tasks.

Popular tools and libraries for computer vision tasks include OpenCV (Open Source Computer Vision Library), TensorFlow, PyTorch, and scikit-image.

Computer vision finds applications in diverse fields such as healthcare (medical image analysis), autonomous vehicles, surveillance, augmented reality, and robotics, among others. The ability to understand and interpret visual information is crucial for many modern technologies and systems.

3. Application of Computer Vision

Computer vision has numerous applications across various industries due to its ability to interpret and understand visual information. Here are some notable applications:

3.1.Healthcare:

Medical Image Analysis: Computer vision is used to analyze medical images (MRI, CT scans, X-rays) for diagnosis and treatment planning.Disease Detection: Detecting diseases and abnormalities in medical images, such as diabetic retinopathy in retinal images.

3.2.Autonomous Vehicles:

Object Detection: Identifying and tracking objects on the road, pedestrians, cyclists, and other vehicles. Lane Detection: Recognizing lane boundaries and road markings for autonomous navigation. Simultaneous Localization and Mapping (SLAM): Building and updating a map of the environment while navigating.

3.3.Retail:

Automated Checkout: Computer vision is used for cashier-less checkout systems where customers can grab items and leave without traditional checkout processes. Inventory Management: Monitoring and managing stock levels on store shelves using visual data.

3.4.Security and Surveillance:

Facial Recognition: Identifying and verifying individuals for access control and security purposes.Anomaly Detection: Detecting unusual behavior or events in surveillance footage.

3.5.Manufacturing:

Quality Control: Inspecting products for defects on production lines using computer vision systems.

Robotic Guidance: Guiding robots in assembly and manufacturing processes based on visual feedback.

3.6.Agriculture:

Crop Monitoring: Analysing drone or satellite images to monitor crop health, detect diseases, and optimize irrigation.

Weed Detection: Identifying and managing weeds in agricultural fields.

3.7.Augmented Reality (AR) and Virtual Reality (VR):

AR Applications: Overlapping computer-generated information onto the real world, enhancing user experiences.

VR Simulation: Creating realistic virtual environments for training and simulations.

3.8.Retail and E-Commerce:

Product Recognition: Allowing users to search for and purchase products by capturing images.

Virtual Try-On: Allowing customers to virtually try on clothing and accessories before purchasing.

3.9. Environmental Monitoring:

Wildlife Monitoring: Tracking and studying wildlife in their natural habitats using cameras.

Air Quality Monitoring: Analyzing images to monitor and assess air quality in urban areas.

3.10. Sports Analytics:

Player Tracking: Analyzing player movements during sports events for performance evaluation.

Instant Replay: Providing instant visual feedback during sports broadcasts.These are just a few examples, and the applications of computer vision continue to expand as technology advances and new use cases are discovered. The ability to extract meaningful information from visual data has transformative effects on various industries.

4. Detection and Enumeration in Computer Vision

Detection and enumeration in computer vision often involve identifying and counting objects or entities within images or video frames. Here's a breakdown of how this process is typically approached:

4.1. Object Detection:

Object detection involves locating and classifying objects within an image. There are various techniques and models used for object detection, including:

4.2. Bounding Box Detection:

Identify objects by drawing bounding boxes around them in an image.

Popular algorithms: YOLO (You Only Look Once), Faster R-CNN (Region-based Convolutional Neural Network), SSD (Single Shot Multibox Detector).

4.3. Mask R-CNN (Instance Segmentation):

Goes beyond bounding boxes by providing pixel-level segmentation of objects in an image.

4.4. Feature-based Detection:

Identify objects based on distinctive features, such as corners or key points.

Algorithms like SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features) can be used.

4.5. Enumeration:

After detecting objects, enumeration involves counting the instances of each object. The process can be as simple as counting the number of bounding boxes

or as complex as analysing the content of each detected object. Here's a general approach:

4.6. Counting Bounding Boxes:

For each detected object, increment a counter. The total count represents the enumeration.

4.7. Region-based Enumeration:

Divide the image into regions of interest and count objects within each region.

4.8. Semantic Enumeration:

If the objects are of different types, count them separately based on their semantic class (e.g., count cars, pedestrians, etc.).

5. Roadmap to the report

The structure of the report is as follows:

Chapter 1: It discusses about brief introduction of what computer vision is, what are there wide applications, some important keywords like detection and counting in computer vision, and finally roadmap of the report.

Chapter 2: It is based on the discussion of one of the domains of computer vision i.e., Human Detection and further is emphasize the detection of humans in real time image, video and through camera and discussed the meaning of Maximum Human Count.

Chapter 3: This chapter is based on key point of various computer vision project or basically any project i.e., Accuracy. Inside that we have discussed about Max. Accuracy we got in detection process, Max. Average Accuracy.

Chapter 4: It talks about the graphical representation results of detection process, based on the data we got. In that we focussed on two basic plot that are Enumeration Plot and Average Accuracy Plot.

Chapter 5: At last, this chapter deals with a brief conclusion and further scope of this project.

Chapter 2

1. Human Detection

- Human detection, also known as pedestrian detection, is a computer vision task that involves identifying and locating humans or pedestrians in images or video frames. This task is crucial in various applications, including surveillance, autonomous vehicles, robotics, and human-computer interaction. Different techniques and models can be used for human detection, ranging from traditional computer vision methods to modern deep learning approaches.
- Human detection is the task of locating all instances of human beings present in an image, and it has been most widely accomplished by searching all locations in the image, at all possible scales, and comparing a small area at each location with known templates or patterns of people.
- Some common methods are :

1.1. Using Haar Cascade Classifier:

Haar cascades are based on Haar-like features and are particularly effective for real-time object detection. Pre-trained Haar cascade classifiers are available for human detection, and OpenCV provides an implementation.

```
import cv2

# Load the pre-trained Haar Cascade classifier for human faces
human_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_fullbody.xml')
# Read the input image
image = cv2.imread('path/to/your/image.jpg')

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Perform human detection using the Haar Cascade classifier
humans = human_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
# Draw rectangles around the detected humans
for (x, y, w, h) in humans:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
# Display the result
cv2.imshow('Human Detection using Haar Cascade', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

1.2. Using HOG(Histogram of Oriented Gradients) :

HOG is a feature descriptor that represents the distribution of gradient orientations in an image. It can be used with machine learning models, such as support vector machines (SVM), for human detection.

- This feature used in computer vision and image processing for the purpose of object detection.
- Along with HOG descriptor we also added the preview button, with which it allows us to see the preview of the image, video selected.
- After detecting, we also printed the max human count in the window.
- And finally we've used a third method using tensorflow which uses deep learning and convolutional neural network.

Figure: 2.1

1.3. Using Tensorflow:

- We implemented the tensorflow method of detecting the human using neural networks.
- And we got better accuracy as compared to the 1st method that was Haar Cascade Classifier and the 2nd method that was using HOG descriptor.
- Both the earlier method, gave almost the same accuracy and only few humans are only getting detected but with tensorflow method, accuracy increased very much and almost all the humans got detected.
- And along with detecting and counting no. of humans, we also got the accuracy of each human with which it is getting detected.

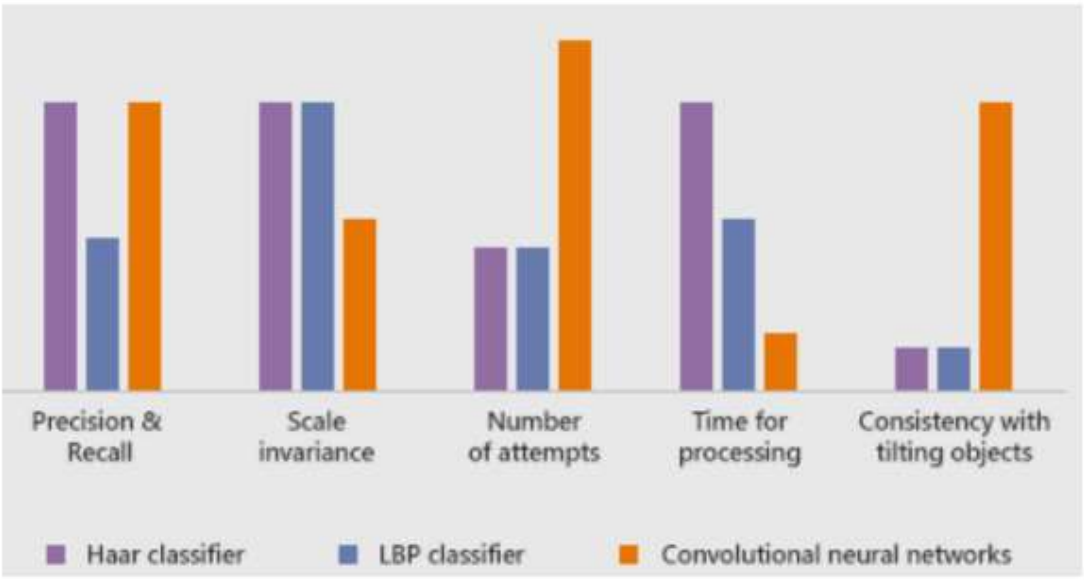


Figure: 2.2

We have implemented the project for the different cases:

- Image
- Video
- Camera

2. Detection & Counting through Image

To perform detection and counting through an image, you can use computer vision techniques and models. In this example, I'll demonstrate how to use a pre-trained object detection model to detect objects in an image and count them. We'll use the YOLO (You Only Look Once) model, which is known for its real-time object detection capabilities.

Here's an example using Python, OpenCV, and a pre-trained YOLO model:

This section works with real time images.

Figure: 2.3

3. Detection & Counting through Video

Detecting and counting objects in videos involves applying object detection techniques to each frame of the video and then aggregating the results. Here's a general outline of the process using a pre-trained object detection model:

Python Example using OpenCV and YOLO:

Assuming you have OpenCV, and the YOLO model files (yolov3.weights, yolov3.cfg, and coco. names for the COCO dataset) available:

This section works with real time videos. Here will allow user to select any real time video from the local system and then user can detect the humans in it.

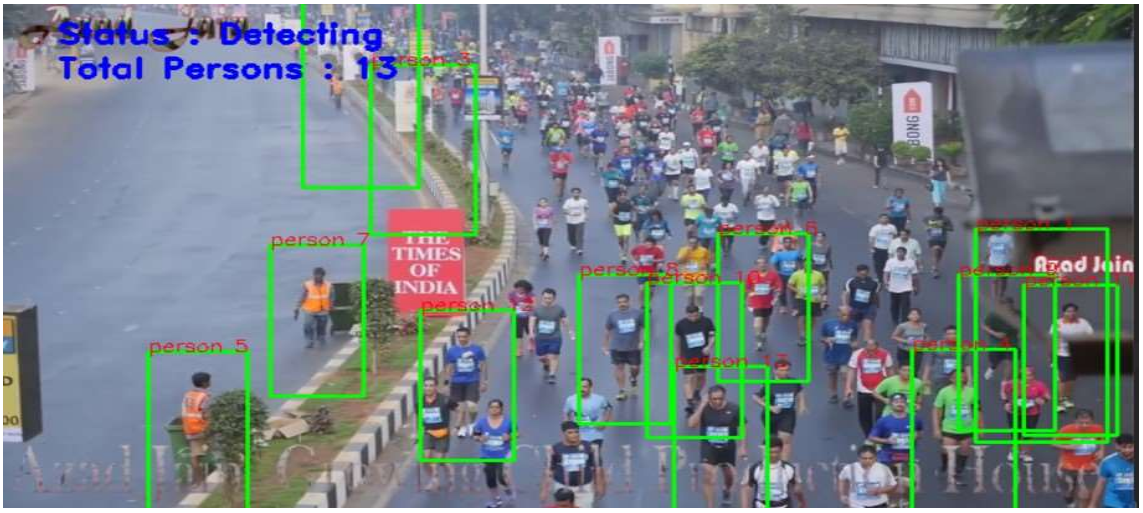


Figure:2.4

```
def detectByPathVideo(path, writer):

    video = cv2.VideoCapture(path)
    check, frame = video.read()
    if check == False:
        print('Video Not Found. Please Enter a Valid Path (Full path of Video Provided).')
        return

    print('Detecting people...')
    while video.isOpened():
        #check is True if reading was successful
        check, frame =  video.read()

        if check:
            frame = imutils.resize(frame , width=min(800,frame.shape[1]))
            frame = detect(frame)

            if writer is not None:
                writer.write(frame)

            key = cv2.waitKey(1)
            if key== ord('q'):
                break
        else:
            break
    video.release()
    cv2.destroyAllWindows()

def detectByCamera(writer):
    video = cv2.VideoCapture(0)
```

```
def detectByCamera(writer):
    video = cv2.VideoCapture(0)
    print('Detecting people...')

    while True:
        check, frame = video.read()

        frame = detect(frame)
        if writer is not None:
            writer.write(frame)

        key = cv2.waitKey(1)
        if key == ord('q'):
                break

    video.release()
    cv2.destroyAllWindows()
```

4. Detection & Counting through Camera

This section works somehow similar to case of video. Here user will be asked to first open the webcam, and it will detect humans that will comes in that webcam during the detection process.

DetectByCamera() method

```
1.  def detectByCamera(writer):
2.      video = cv2.VideoCapture(0)
3.      print('Detecting people...')
4.
5.      while True:
6.          check, frame = video.read()
7.
8.          frame = detect(frame)
9.          if writer is not None:
10.             writer.write(frame)
11.
12.         key = cv2.waitKey(1)
13.         if key == ord('q'):
14.             break
15.
16.     video.release()
17.     cv2.destroyAllWindows()
```

cv2.VideoCapture(0) passing 0 in this function means we want to record from a webcam. **video. Read ()** read frame by frame. It returns a check which is True if this was able to read a frame otherwise False.

Now, For each Frame, we will call detect() method. Then we write the frame in our output file.

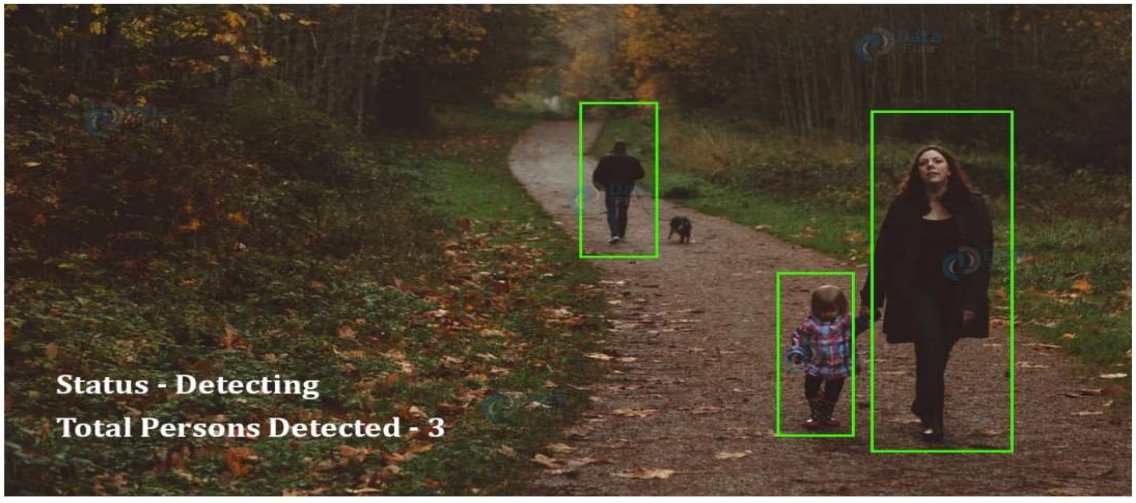


Figure: 2.5

Chapter: 3

1. Graphical user interface

- Added frontend part in the project. For this we used Tkinter GUI and added different buttons and labels.
- Additional Libraries used:
  - Tkinter – for frontend GUI window
  - PIL – for adding images to the tkinter GUI window
  - Messagebox from tkinter – to show any message using dialog
  - Filedialog – to select images and video using select buttons.
- When we run the code, a GUI window will open with START and EXIT button on it.

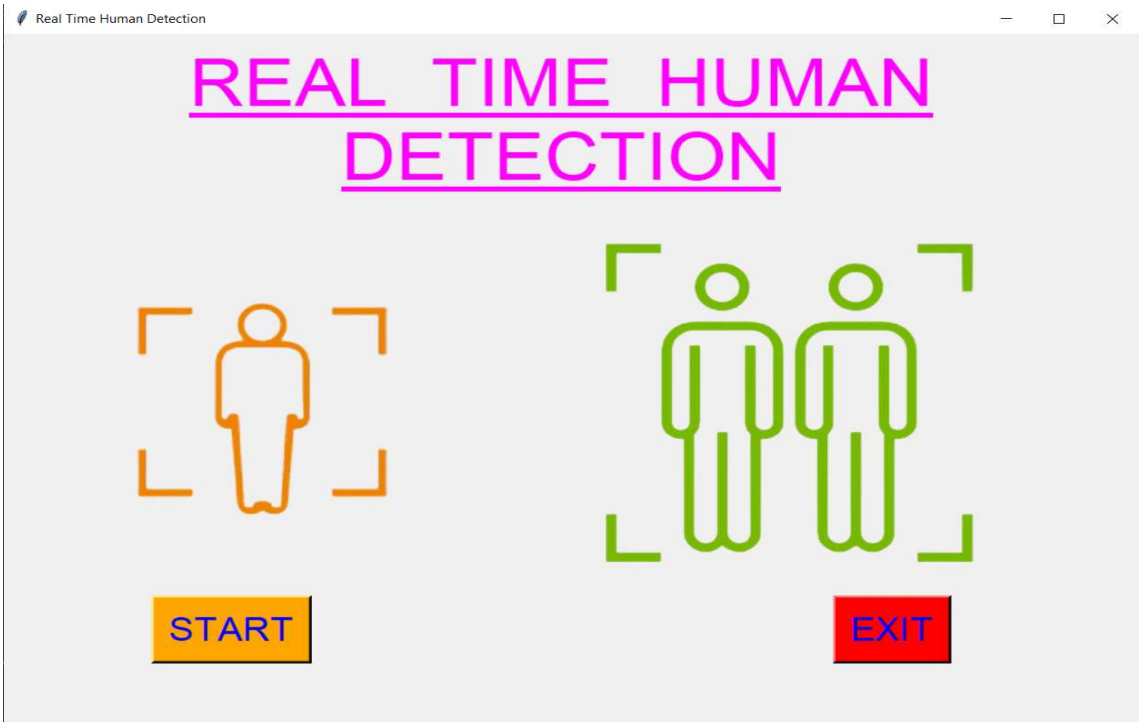
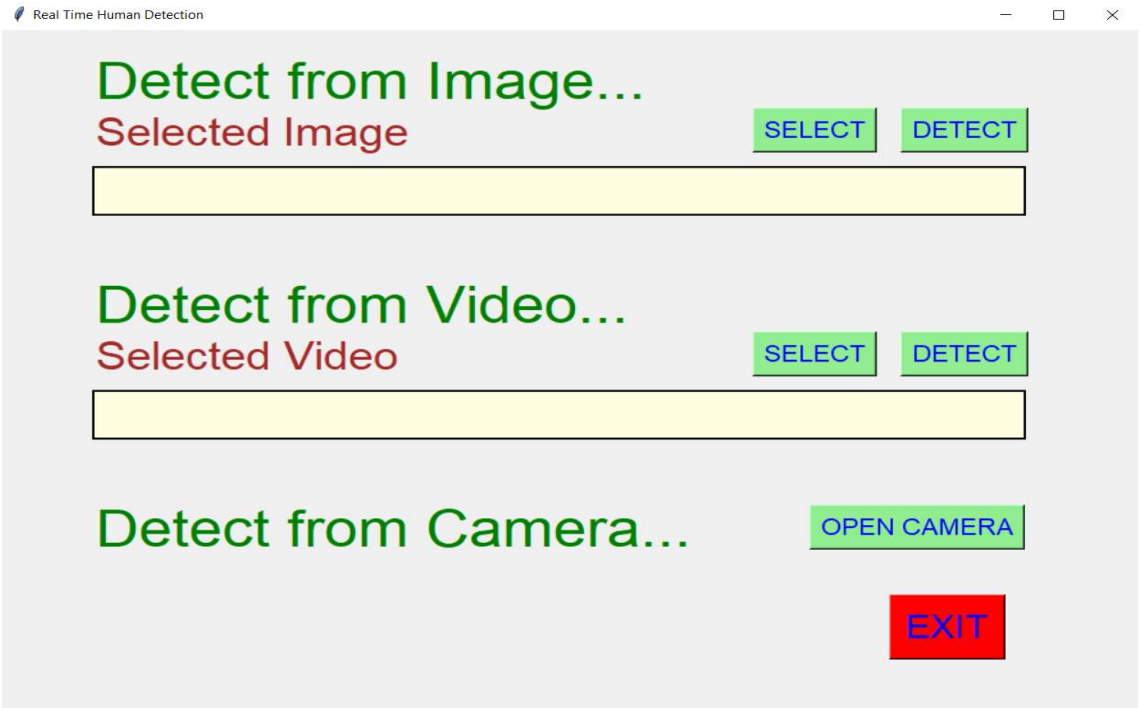


Figure:3.1



sFigure:3.2

- When we click on select button on image option, we get option to select image from the local system and when click on the detect button, an output image is shown which detected human and their count

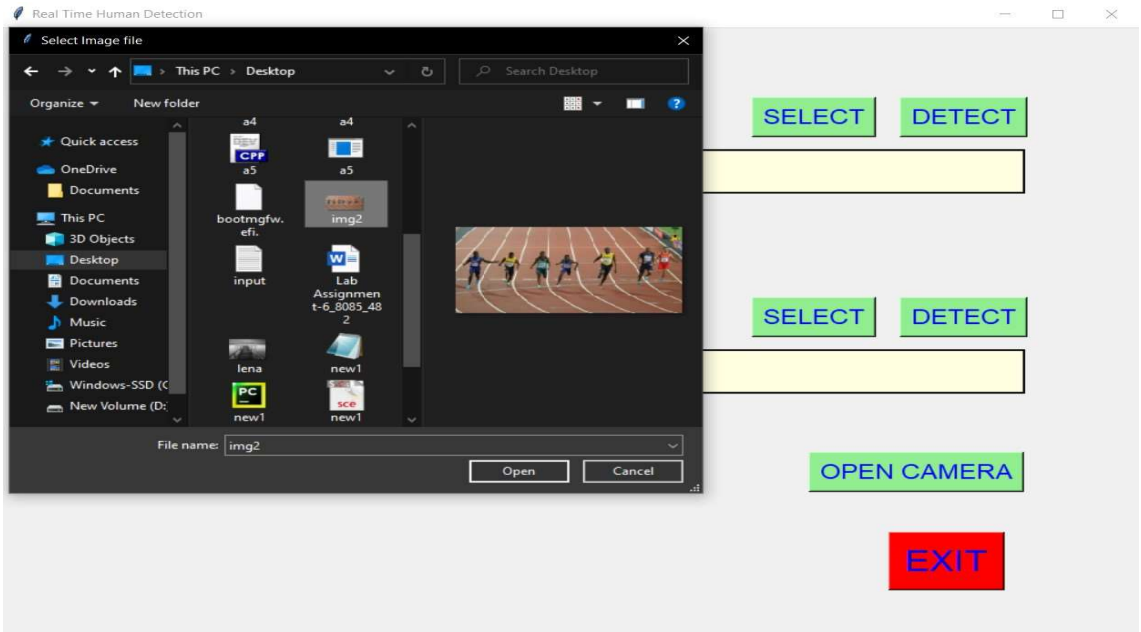


Figure:3.3

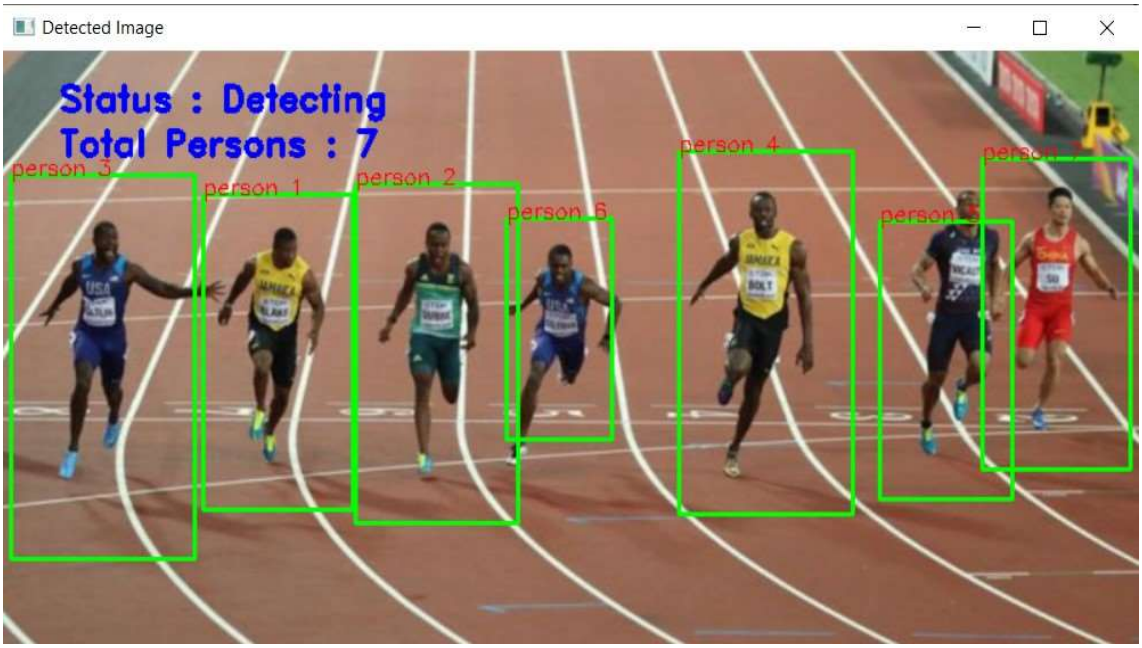


Figure:3.4

- In the same way when we select video from the local system, it will give the output by detecting the humans in the video.

2. CNN – Convolutional Neural network

A Convolutional Neural Network (CNN) is a type of deep neural network specifically designed for tasks involving images, although it can also be applied to other grid-like data. CNNs have proven to be highly effective in computer vision tasks such as image classification, object detection, and image segmentation.

3. Key Components of CNN:

3.1. Convolutional Layers:

Convolutional layers are the core building blocks of a CNN. They consist of filters (also called kernels) that slide over the input image to perform convolution operations. These operations capture spatial hierarchies of features.

3.2. Activation Functions:

Activation functions, such as ReLU (Rectified Linear Unit), are applied element-wise to the output of convolutional layers, introducing non-linearity to the network.

3.3. Pooling (Subsampling) Layers:

Pooling layers are used to reduce the spatial dimensions of the input volume, effectively down sampling the feature maps. Max pooling is a common pooling operation.

3.4. Fully Connected (Dense) Layers:

Fully connected layers are traditional neural network layers where each neuron is connected to every neuron in the previous and next layers. These are often used in the final layers of the network for classification tasks.

Flattening:Before connecting to fully connected layers, the high-level reasoning in the convolutional and pooling layers is flattened into a vector.

3.5. CNN Architecture:

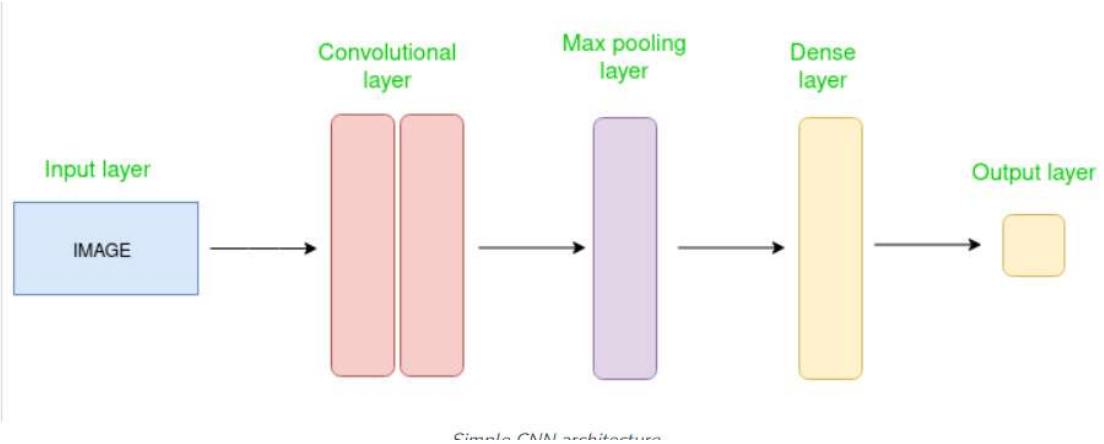
The typical architecture of a CNN consists of alternating convolutional and pooling layers followed by one or more fully connected layers. A common architecture is as follows:

Input Layer
Convolutional Layer + ReLU Activation
Pooling Layer
Convolutional Layer + ReLU Activation
Pooling Layer Flattening
Fully Connected (Dense) Layer + ReLU Activation
Output Layer (Softmax Activation for multi-class classification)
Training a CNN:

Training a CNN involves providing it with labeled training data and using backpropagation to update the weights and biases of the network to minimize the difference between predicted and actual labels.

Example Implementation using TensorFlow and Keras:

Here's a simplified example of a CNN for image classification using TensorFlow



Step

- import the necessary libraries
- set the parameter
- define the kernel
- Load the image and plot it.
- Reformat the image
- Apply convolution layer operation and plot the output image.
- Apply activation layer operation and plot the output image.
- Apply pooling layer operation and plot the output image**.**

```
# import the necessary libraries
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from itertools import product

# set the param
plt.rc('figure', autolayout=True)
plt.rc('image', cmap='magma')

# define the kernel
kernel = tf.constant([[-1, -1, -1],
                      [-1,  8, -1],
                      [-1, -1, -1],
                     ])

# load the image
image = tf.io.read_file('Ganesh.jpg')
image = tf.io.decode_jpeg(image, channels=1)
image = tf.image.resize(image, size=[300, 300])

# plot the image
img = tf.squeeze(image).numpy()
plt.figure(figsize=(5, 5))
plt.imshow(img, cmap='gray')
plt.axis('off')
plt.title('Original Gray Scale image')
plt.show();
```

OUTPUT

Figure:3.7

4. HOG – Histogram of Oriented Gradients

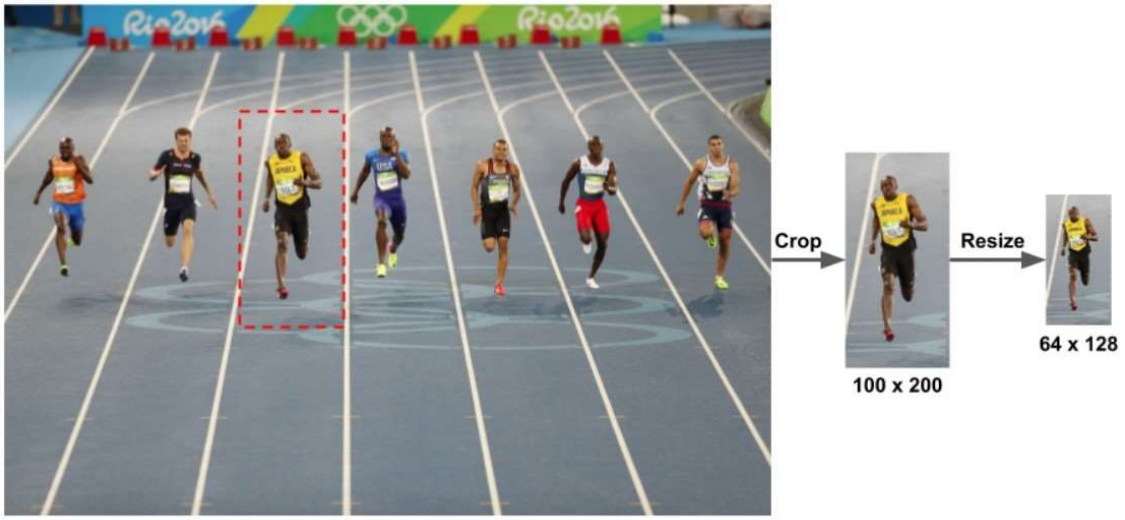
The histogram of oriented gradients (HOG) is a used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localize portions of an image.

How to calculate Histogram of Oriented Gradients

Step1: Preprocessing

As mentioned earlier HOG feature descriptor used for pedestrian detection is calculated on a 64×128 patch of an image. Of course, an image may be of any size. Typically patches at multiple scales are analysed at many image locations. The only constraint is that the patches being analysed have a fixed aspect ratio. In our case, the patches need to have an aspect ratio of 1:2. For example, they can be 100×200, 128×256, or 1000×2000 but not 101×205.

To illustrate this point I have shown a large image of size 720×475. We have selected a patch of size 100×200 for calculating our HOG feature descriptor. This patch is cropped out of an image and resized to 64×128. Now we are ready to calculate the HOG descriptor for this image patch.



Step 2: Calculate the Gradient Images

To calculate a HOG descriptor, we need to first calculate the horizontal and vertical gradients; after all, we want to calculate the histogram of gradients. This is easily achieved by filtering the image with the following kernels.

-1	0	1

-1
0
1

Next, we can find the magnitude and direction of gradient using the following formula

$$g = \sqrt{g_x^2 + g_y^2}$$
$$\theta = \arctan \frac{g_y}{g_x}$$

If you are using OpenCV, the calculation can be done using the function **cartToPolar** as shown below.

```
// C++ Calculate gradient magnitude and direction (in degrees)
Mat mag, angle;
cartToPolar(gx, gy, mag, angle, 1);
```

The same code in python looks like this.

```
# Python Calculate gradient magnitude and direction ( in degrees )
mag, angle = cv2.cartToPolar(gx, gy, angleInDegrees=True)
```

The figure below shows the gradients.

Left : Absolute value of x-gradient. Center : Absolute value of y-gradient.

Right : Magnitude of gradient.

Notice, the x-gradient fires on vertical lines and the y-gradient fires on horizontal lines. The magnitude of gradient fires where ever there is a sharp change in intensity. None of them fire when the region is smooth. I have deliberately left out the image showing the direction of gradient because direction shown as an image does not convey much. The gradient image removed a lot of non-essential information (e.g. constant coloured background), but highlighted outlines. In

other words, you can look at the gradient image and still easily say there is a person in the picture.
At every pixel, the gradient has a magnitude and a direction. For color images, the gradients of the three channels are evaluated (as shown in the figure above). The magnitude of gradient at a pixel is the maximum of the magnitude of gradients of the three channels, and the angle is the angle corresponding to the maximum gradient.

Chapter 4

1. Accuracy

Accuracy is a commonly used metric to evaluate the performance of a model in computer vision tasks, especially in classification problems. It measures the percentage of correctly predicted instances out of the total instances.

The formula for accuracy is:

Accuracy =Number of Correct Predictions/ Total Number of Predictions

In the context of computer vision, where tasks like image classification are common, accuracy is often used to assess how well a model can correctly classify images into different categories.

Here's a breakdown of key terms related to accuracy in computer vision:

Number of Correct Predictions:The count of instances where the model correctly predicts the true class of the input.

Total Number of Predictions:The total count of instances for which predictions are made.

Example:

Suppose you have a classification model for recognizing handwritten digits (0 to 9) using the MNIST dataset. After training the model, you evaluate its performance on a test set of 1,000 images.

The model correctly predicts the digit in 900 images.

The total number of predictions is 1,000.

Accuracy = 900/1000

= 0.9

So, in this example, the accuracy of the model is 90%

Considerations:

Imbalanced Datasets:

Accuracy might not be the best metric for highly imbalanced datasets, where one class dominates the others. In such cases, other metrics like precision, recall, or F1-score may provide a more comprehensive evaluation.

Confusion Matrix:

Analyzing a confusion matrix (which shows true positives, true negatives, false positives, and false negatives) alongside accuracy can provide more insights, especially in multi-class classification scenarios.

Task-Specific Metrics:

Depending on the specific computer vision task, additional metrics might be more relevant. For instance, in object detection, metrics like Intersection over Union (IoU) or mAP (mean Average Precision) are commonly used.

It's essential to choose evaluation metrics based on the characteristics of your dataset and the goals of your specific computer vision task. While accuracy is a widely used metric, it's crucial to consider the context and potential limitations of using it alone

2. Maximum Accuracy

Achieving the maximum accuracy in a Convolutional Neural Network (CNN) depends on several factors, including the complexity of the task, the quality and quantity of the training data, the architecture of the CNN, and the training parameters. Here are some tips to maximize accuracy:.

3. Maximum Average Accuracy

To achieve the maximum average accuracy in a Convolutional Neural Network (CNN), you should consider several factors and follow best practices in the design, training, and evaluation of your model. Here are some tips to help you achieve high accuracy:

3.1. Quality of Data:

Ensure that your dataset is clean, well-labeled, and representative of the real-world scenarios you want your model to perform well on.

3.2. Data Augmentation:

Use data augmentation techniques to artificially increase the size of your training dataset. Techniques like rotation, flipping, and zooming can help the model generalize better to variations in the input data.

3.3. Normalization:

Normalize your input data to bring it to a similar scale. This helps in faster convergence during training.

3.4. Appropriate Architecture:

Design a CNN architecture that is appropriate for your task. Consider the complexity of your problem and choose an architecture with an adequate number of layers, filters, and neurons.

3.5. Regularization:

Use regularization techniques such as dropout or L2 regularization to prevent overfitting, especially if you have a limited amount of training data.

3.6. Learning Rate:

Experiment with different learning rates during training. Too high of a learning rate can cause the model to diverge, while too low of a learning rate may result in slow convergence.

Chapter: 5

1. Plots

This section basically deals with the graphical representation of the data we got from the detection process. Using this graphical representation, one can do the analysis of the human count and accuracy very well.

In our application, we have basically talked about two basic plots.

o Enumeration Plot

o Avg. Accuracy Plot

2. Enumeration Plot

This plot basically represents the plot between humans count against each time interval. For this plot, the parameter we took on X-axis is time (in seconds) and on Y-axis, we took, human count at that particular time.

And the highest peak in this enumeration plot, indicates the maximum no. of people detected in whole detection process.

```
import matplotlib.pyplot as plt

# Example data - replace this with your actual data
categories = ['Category A', 'Category B', 'Category C']
counts = [15, 30, 20]

# Create a bar plot
plt.bar(categories, counts, color='skyblue')

# Add labels and title
plt.xlabel('Categories')
plt.ylabel('Counts')
plt.title('Enumeration Plot')

# Display the plot
plt.show()
```

Figure: 5.1

3. Average Accuracy Plot

Creating average accuracy plots involves tracking the accuracy of a machine learning model over time or different settings. Here is a generic example of how you might create an average accuracy plot using Python and the popular matplotlib library. Please note that this is a conceptual example, and you would

```python
import matplotlib.pyplot as plt
import numpy as np

# Example data: epochs and corresponding accuracy values
epochs = np.arange(1, 11)  # replace with your actual epoch values
accuracy_values = np.random.rand(10)  # replace with your actual accura

# Compute the running average
running_average = np.cumsum(accuracy_values) / np.arange(1, len(accurac

# Plot the accuracy values
plt.plot(epochs, accuracy_values, label='Accuracy')
plt.plot(epochs, running_average, label='Running Average', linestyle='-

# Add labels and title
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
```

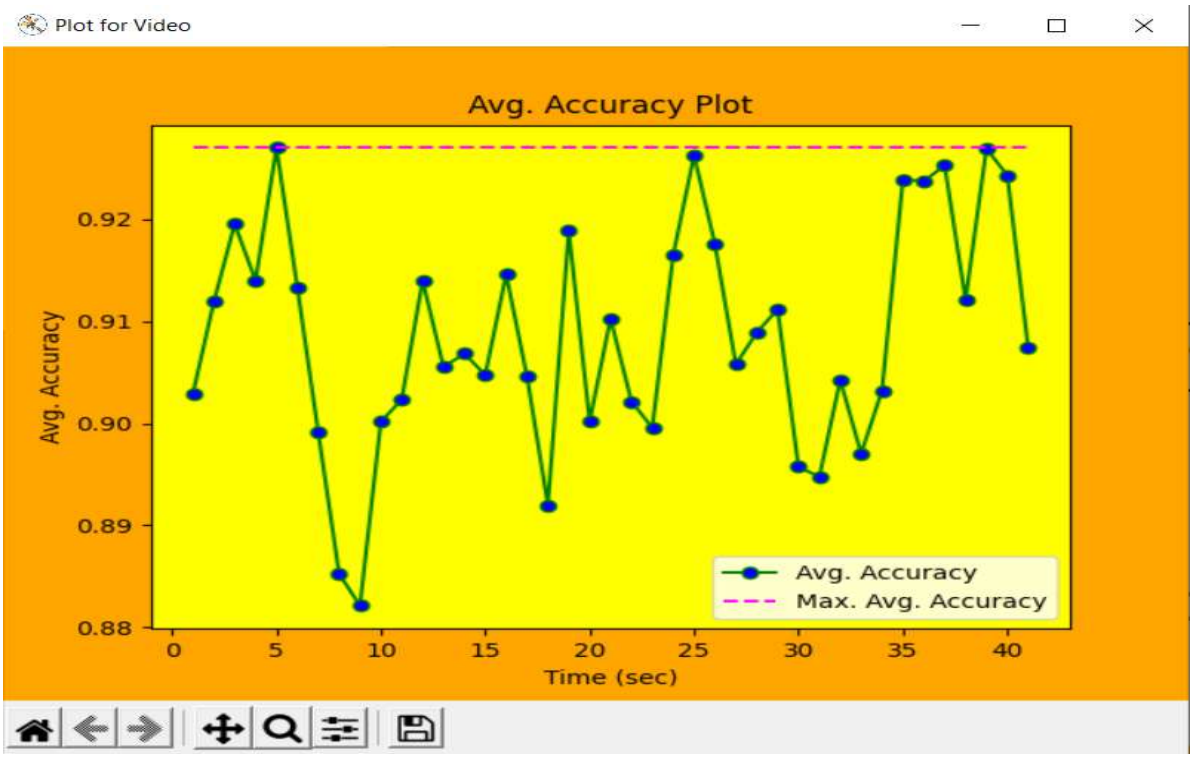


Figure: 5.2

Chapter 6

Conclusion and Future Scopes

In the last section of the project, we generate Crowd Report [5], which will give some message based on the results we got from the detection process. For this we took some threshold human count, and we gave different message for different results of human count we got form detection process.

Now coming to the future scope of this project or application, since in this we are taking any image, video or with camera we are detecting humans and getting count of it, along with accuracy. So some of the future scope can be:

- This can be used in various malls and other areas, to analyse the maximum people count, and then providing some restrictions on number of people to have at a time at that place.
- This can replace various mental jobs, and this can be done more efficiently with machines.
- This will ultimately lead to some kind of crowd-ness control in some places or areas when implemented in that area.

.

Bibliography

[1] Programming Computer Vision with Python, 1st Edition, Jan Eric Solem, 2012, O' Reily

[2] Learning OpenCV, Adrian Kaehler and Gary Rost Bradski, 2008, O' Reily

[3] Deep Learning with TensorFlow, Giancarlo Zaccone, Md. Rezaul Karim, Ahmed Menshawy, 2017

[4] Python GUI Programming with Tkinter, Alan D. Moore, 2018

[5] Python Standard Library, Fredrik Lundh, 2001, O' Reil

Website

https://learnopencv.com/histogram-of-oriented-gradients.
https://www.geeksforgeeks.org/introduction-convolution-neural-network/
https://data-flair.training/blogs/python-project-real-time-human-detection-counting/

PLAGIARISM REPORT