FLOATING MARINE LITTER DETECTION WITH EDGE COMPUTING AND LOCATION ACQUISITION

A project report submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Electronics and Communication Engineering

By

RASAPUTRA MOUNIKA (620231)

MALLULA JAHNAVI (620208)

EARABOINA RISHI VARDHAN (620128)

Department of Electronics and Communication Engineering

NATIONAL INSTITUTE OF TECHNOLOGY
ANDHRA PRADESH-534101

APRIL 2024

BONAFIDE CERTIFICATE

This is to certify that the project titled **FLOATING MARINE LITTER DETECTION WITH EDGE COMPUTING AND LOCATION ACQUISITION** is a bonafide record of the work done by

RASAPUTRA MOUNIKA (620231)

MALLULA JAHNAVI (620208)

EARABOINA RISHI VARDHAN (620128)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **ECE** of the **NATIONAL INSTITUTE OF TECHNOLOGY, ANDHRA PRADESH**, during the year 2023-2024.

Mr. Kondala Rao Jyothi	**Dr.S.Yuvaraj**
Project Guide	Head of the Department

ABSTRACT

Marine litter is a persistent challenge threatening aquatic ecosystems, demands urgent attention and innovative solutions to mitigate its effects efficiently. Traditional cleaning operations, reliant on human efforts, are labor-intensive and time-consuming. Thus, there's an imperative need to automate marine litter detection processes, aiming to reduce human intervention and expedite response measures.

This thesis proposes a comprehensive automated system leveraging technologies to address the challenges posed by marine pollution. By integrating multiple image processing and machine learning algorithms, the system aims to autonomously detect marine litter in images captured by drone. Furthermore, the integration of GPS-based geolocation facilitates precise mapping of detected marine litter, enabling targeted cleanup efforts and effective resource allocation. Implementing the entire system on a Raspberry Pi platform ensures real-time application, enabling prompt responses to emerging pollution incidents.

By reducing reliance on manual surveys and streamlining detection processes, the proposed system not only optimizes resource utilization but also enables timely interventions to mitigate environmental damage. Ultimately, this research contributes to advancing the field of marine litter detection and management, paving the way for more efficient and sustainable solutions to combat ocean pollution. Through automation and technological innovation, the proposed system strives to safeguard aquatic ecosystems and preserve biodiversity for future generations.

Keywords: Litter cleaning,Machine Learning,Geolocation,Raspberry pi,Automation

ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

Mr. Kondala Rao Jyothi, our project guide,for his continuous support and motivation throughout the project work. His advices and guidelines at every stage have helped us to tackle the challenges we faced during the work.The experience we gained working under his guidance helped us to excel in our field of research.

We profoundly thank **Dr. S. Yuvaraj**, the Head of the Department ECE who has been an excellent guide and also a great source of inspiration to our work.

Our internal reviewers, **Dr.P Kishore Kumar**, **Dr.M Ananda Reddy**, **Dr.B Thulasya Naik** for their insight and advice provided during the review sessions. We would also like to thank our individual parents and friends for their constant support.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 INTRODUCTION

Marine litter, an escalating environmental concern, poses a profound threat to the health and integrity of aquatic ecosystems worldwide. Comprising an array of discarded materials, including plastics, debris, and other waste, its pervasive presence necessitates urgent and effective intervention strategies. Traditional methods of marine litter detection and cleanup often prove inadequate, demanding innovative approaches to comprehensively address the issue. This thesis endeavours to advance the field by proposing a holistic system for marine litter detection and cleanup, integrating cutting-edge technologies and methodologies. The proposed system combines several advanced techniques, including Canny edge detection, adaptive thresholding, K-means clustering, CNNs, HOG+SVM, and the YOLOv8 detection model, to enhance the efficiency and accuracy of marine litter detection processes. Canny edge detection and adaptive thresholding are employed to identify potential regions of interest within imagery captured by drones, effectively delineating boundaries of marine litter items against dynamic backgrounds. Subsequently, K-means clustering is utilized to segment and classify individual litter objects based on their visual characteristics, such as shape, color, and texture. To further augment detection capabilities, the system integrates convolutional neural networks (CNNs) trained on labeled datasets of marine litter images. These CNNs leverage deep learning algorithms to extract intricate features from images, facilitating precise classification and categorization of marine litter items. Additionally, the system employs Histogram of Oriented Gradients (HOG) combined with Support

Vector Machines (SVM) for object detection, enhancing detection accuracy and robustness. Beyond detection, the system incorporates automated litter pickup functionalities facilitated by the YOLOv8 detection model. Leveraging real-time object detection capabilities, the system autonomously identifies and localizes marine litter items within imagery captured by drones. This information guides the deployment of robotic or unmanned aerial vehicle (UAV) platforms equipped with retrieval mechanisms, facilitating the collection and removal of detected litter items from marine environments. Moreover, the entire system is implemented on a Raspberry Pi 4 platform, enabling real-time application and on-the-fly processing of drone-captured imagery. The incorporation of a GPS module facilitates geolocation tagging, enabling precise mapping of detected marine litter and enhancing the system's effectiveness in targeted cleanup efforts.[1]

1.2 EXISTING WORK

1.2.1 Using drones and satellite images

Targets made from plastic bottles, bags, and fishing nets were analyzed using drone and Sentinel-2 data. Sentinel-2 showed promise, detecting plastics over large areas despite image resolution limitations. NIR bands were particularly useful, aiding in detection.[2]

Figure 1.1: Images captured by drone and satellites

1.2.2 Sensors

The water surface cleaning robots use a variety of sensors to detect litter. Here are the sensors used Ultrasonic Distance-Measuring Sensor, Pressure Sensor, Visual Sensor, Power Module , Wireless Communication Module, GPS, Vision Sensor, Attitude Sensor, Wind Direction Sensor, Proximity Sensors, Ultrasonic Sensor, IR Sensor.[3] These sensors play a crucial role in enabling the robots to detect floating objects, sense the environment, identify garbage, and navigate autonomously on the water surface for effective cleaning.[4]

Figure 1.2: Types of sensors

1.2.3 Convolutional Neural Network

The algorithm is based on a deep learning approach that uses convolutional neural networks (CNNs) capable of learning from unstructured or unlabelled data. The CNN-based deep learning model was trained and tested using 3723 aerial images (50 percent containing FMML, 50 percent without FMML) taken by drones and aircraft.[5]

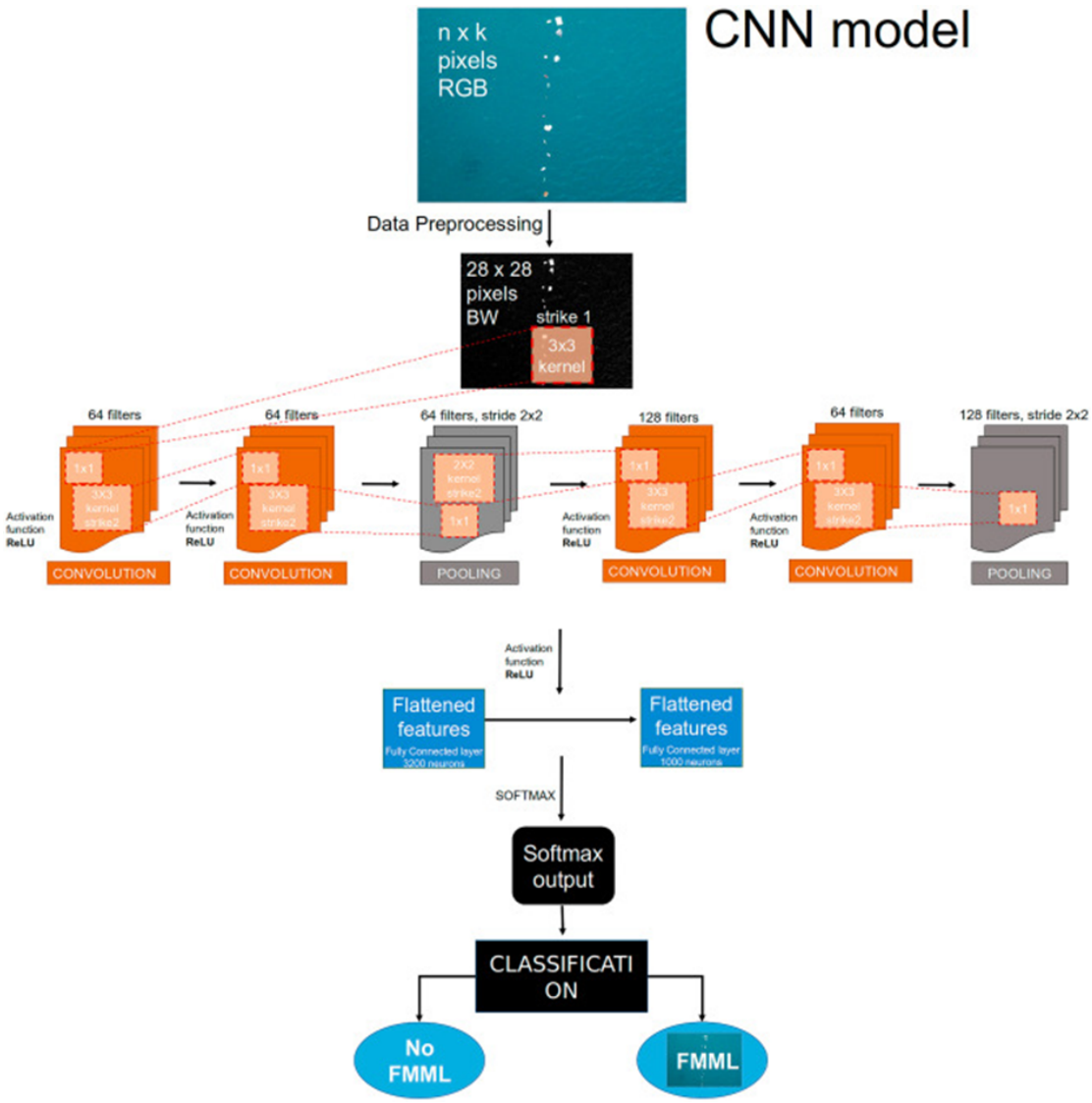


Figure 1.3: CNN Architecture

1.3 Motivation

Our project is driven by the critical need to combat the escalating environmental crisis posed by marine litter. Conventional approaches to detection and cleanup have proven insufficient, demanding innovative solutions. By harnessing advanced technologies such as image processing, machine learning, and robotics, the project aims to revolutionize marine litter management. The integration of these cutting-edge tools promises to enhance detection accuracy, streamline cleanup operations, and reduce the environmental and economic toll of marine pollution. Ultimately, the project seeks to safeguard the health of aquatic ecosystems, protect marine life, and ensure the well-being of coastal communities worldwide. By addressing the root causes of marine litter and implementing effective intervention strategies, this endeavor strives to pave the way towards a more sustainable future for our oceans and planet as a whole.

1.4 Problem statement

Despite the increasing global awareness of the detrimental effects of marine litter on aquatic ecosystems, traditional methods of detection and cleanup have proven insufficient in addressing this escalating environmental crisis. Conventional approaches lack the efficiency and accuracy required to comprehensively tackle the pervasive presence of marine litter. Furthermore, the manual nature of existing cleanup efforts is both labor-intensive and time-consuming, hindering effective mitigation of marine pollution. To address these challenges, there is an urgent need for innovative solutions that leverage advanced technologies such as image processing, machine learning, and robotics. By developing a holistic system for marine litter detection and cleanup, integrating cutting-edge methodologies and tools, this project aims to revolutionize current practices and mitigate the adverse impacts of marine pollution on aquatic ecosystems and coastal communities.

Chapter 2

LITERATURE REVIEW

[6] Odei Garcia Garin and Toni Monleon-Getino are the authors of the paper Automatic detection and quantification of floating marine macro-litter in aerial images. It introduces advanced computer vision techniques for floating marine macro-litter (FML) detection in aerial images, accurately identifying potential FML objects by segmenting and localizing them within the images. Following detection, FML objects undergo classification using machine learning algorithms like Support Vector Machines (SVM), Random Forests, or Convolutional Neural Networks (CNNs). These algorithms leverage features such as shape, color, and texture to precisely categorize FML based on type and size. Trained on annotated datasets, these techniques enable automated detection and qualification of FML, streamlining marine litter monitoring and management processes. By combining these advanced computer vision and machine learning approaches, the proposed system offers a robust solution for efficient FML detection and classification in aerial imagery, contributing to the conservation of marine ecosystems and addressing environmental concerns related to marine litter.

[7] In their extensive literature survey within the 2023 paper, "Real-Time Instance Segmentation for Detection of Underwater Litter as a Plastic Source," Brendan Chongzhi Corrigan, Z. Y. Tay, and D. Konovessis investigate the utilization of neural network models for ocean plastic detection. Their review encompasses training YOLACT and Mask R-CNN models for instance segmentation of underwater litter, drawing insights from the TrashCAN dataset and pre-trained COCO weights. Notably, their survey reveals that YOLACT achieves a mean average precision (mAP) of 0.365, slightly trail-

ing Mask R-CNN's 0.377. Yet, the standout observation lies in the performance-speed trade-off, where YOLACT surpasses Mask R-CNN by detecting images up to six times faster while maintaining comparable accuracy. This analysis positions YOLACT as an efficient solution for real-time litter collection tasks by autonomous underwater vehicles (AUVs), complemented by Mask R-CNN's suitability for detailed litter distribution surveys. Overall, their comprehensive literature survey underscores the promising role of neural network-based approaches in advancing the real-time detection and management of underwater litter, contributing substantially to combatting plastic pollution in marine environments.

[8] In their paper titled "Real-Time Flying Object Detection with YOLOv8," Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi present an innovative method for real-time detection of flying objects using YOLOv8, a cutting-edge single-shot detector. Trained on a diverse dataset encompassing 40 distinct classes of flying objects, the model excels at extracting abstract feature representations. Leveraging transfer learning on a realistic dataset representative of real-world environments, the authors effectively address challenges such as spatial variations, occlusion, and complex backgrounds. The refined model achieves an impressive mean average precision (mAP50-95) of 0.835 while maintaining an average inference speed of 50 frames per second on 1080p videos, showcasing its efficacy in real-time applications. Despite the absence of an official paper specifically dedicated to YOLOv8, the authors meticulously elucidate its architecture and functionality, contributing significantly to the fields of computer vision and pattern recognition. This research offers invaluable insights for both academic exploration and practical implementation in various domains, highlighting YOLOv8's transformative potential in revolutionizing object detection methodologies for real-time flying object detection.

[9] The paper authored by Matshehla Konaite, Pius A Owolawi, Temitope Mapayi, Vusi Malele, Kehinde Odeyemi, Gbolahan Aiyetoro, and Joseph S. Ojo addresses the critical need to enhance navigation and awareness for visually impaired individuals by developing a system capable of real-time object and road/traffic sign detection. Leveraging

the Single-Shot Multibox Detector (SSD) MobileNet v2 convolutional neural network on a Raspberry Pi 4 with TensorFlow Lite 2, the proposed system offers lightweight and efficient assistance to visually impaired users. Technical details highlight the utilization of MobileNet's 3x3 depthwise separable convolutions, optimizing computing resources without compromising accuracy. The SSD MobileNet v2 320x320 model, trained on the COCO dataset, enables object detection, while traffic light sign classification is integrated using machine learning techniques. Although lacking detailed quantitative evaluation metrics, the system demonstrates a real-time performance of approximately 5 frames per second on the Raspberry Pi 4, deemed adequate for practical use. This research holds significance in augmenting the quality of life for visually impaired individuals by providing enhanced navigation and environmental awareness capabilities. Moreover, its deployment on resource-constrained embedded platforms like Raspberry Pi showcases the feasibility of assistive technology implementation in real-world scenarios, making it a valuable contribution to the field of assistive technology and computer vision.

Chapter 3

PROPOSED WORK WITH ITS METHODOLOGY

3.1 Introduction

Marine debris, particularly plastic waste, poses a severe threat to marine ecosystems, necessitating urgent action. Our proposed multi-modal image processing approach integrates various techniques, including Canny edges, k-means clustering, Histogram of Oriented Gradients (HOG), Convolutional Neural Networks (CNNs), adaptive thresholding, contour analysis, Support Vector Machine (SVM), Random Forest, and YOLOv8, to enhance debris detection capabilities. For real-time application, we investigate two distinct strategies: HOG combined with SVM for object detection and YOLOv8 for classification. Additionally, we incorporate a GPS module to ensure precise location tracking of detected debris. By synergizing traditional computer vision methods with deep learning, our approach enables efficient detection and recognition of marine debris. Operating on edge devices, this strategy facilitates swift and accurate monitoring, thereby making significant contributions to marine conservation efforts. Our initiative underscores the importance of leveraging advanced technology to combat the detrimental effects of marine debris and safeguard the health and biodiversity of our oceans.[10]

3.2 Canny Edge Detection

Canny edge detection is a popular image processing technique that can accurately detect edges in images, including marine debris edges. the canny algorithm involves steps

such as gaussian smoothing, gradient computation, non-maximum suppression, thresholding, and edge tracking by hysteresis.The Canny edge detection algorithm involves mathematical operations to compute gradients and perform edge thinning. Here's a mathematical derivation of the main steps:

1.Gradient Calculation:

- The gradient of an image represents the rate of change of intensity at each pixel.The Sobel operator is commonly used to compute gradients in the horizontal and vertical directions.
- Let's denote the input image as I and the gradients in the horizontal and vertical directions as G_x and G_y respectively.
- The Sobel operator kernels for gradient calculation are:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The convolution of the input image with these kernels gives the horizontal and vertical gradients:

$$G_x = I * G_x$$

$$G_y = I * G_y$$

The gradient magnitude M and direction θ can then be calculated as follows:

$$M(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2}$$

$$\theta(x, y) = \arctan \frac{G_y(x, y)}{G_x(x, y)}$$

2. Edge Thinning:

- After obtaining the gradient magnitude and direction, the algorithm performs edge thinning to obtain thin edge contours.

- This is achieved through non-maximum suppression.For each pixel in the gradient magnitude image, compare its magnitude with the magnitudes of its neighbors in the direction perpendicular to the edge.

- If the magnitude of the pixel is not greater than both of its neighbors, set its value to zero.This step effectively preserves only local maxima along edges.Mathematically, this can be represented as:

$$M_{\text{thin}}(x,y) = \begin{cases} M(x,y) & \text{if } M(x,y) \text{ is the local maximum in the direction of } \theta(x,y) \\ 0 & \text{otherwise} \end{cases}$$

where M_{thin} is the thinned gradient magnitude image.
These mathematical operations form the core of the Canny edge detection algorithm. After these steps, thresholds are applied to determine which edges are significant and which are not, followed by edge linking to connect adjacent edges into continuous contours.[11]

3.3 K-means clustering

In the context of marine debris detection, k-means clustering can be used to segment images or data points into clusters that potentially represent debris objects based on their visual characteristics, such as color, texture, and shape. By applying k-means clustering to marine debris images or data points, we can obtain segmented regions that may contain debris objects, which can be further processed for classification or further analysis, such as feature extraction or object recognition.K-means clustering is a popular unsupervised machine learning algorithm. The algorithm iteratively assigns each data point to the nearest centroid (representative point) and then recalculates the centroids based on the mean of the points assigned to each cluster. Here's a mathematical derivation of the K-means clustering algorithm:

Let $X = \{x_1, x_2, ..., x_n\}$ be a dataset consisting of n data points in d-dimensional space.

1. Initialization:

   - Choose k initial centroids $\{c_1, c_2, ..., c_k\}$ randomly from the dataset.

2. Assignment Step:

   - For each data point x_i, calculate its distance to each centroid c_j using a distance metric (usually Euclidean distance):

$$\text{dist}(x_i, c_j) = \sqrt{\sum_{l=1}^{d} (x_{il} - c_{jl})^2}$$

   - Assign each data point x_i to the cluster associated with the nearest centroid c_j. Mathematically, this can be represented as:

$$\text{Cluster}(x_i) = \text{argmin}_j \text{dist}(x_i, c_j)$$

3. Update Step:

   - After assigning all data points to clusters, update the centroids of the clusters based on the mean of the points assigned to each cluster.for each cluster C_j, calculate the new centroid c_j as the mean of all data points assigned to that cluster:

$$c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

4. Convergence:

   - Repeat the assignment and update steps iteratively until convergence criteria are met.

   - Convergence can be defined by the centroids no longer changing significantly or a maximum number of iterations being reached.

5. Output:

   - The final output of the algorithm is the set of centroids $\{c_1, c_2, ..., c_k\}$ and the assignment of each data point to a cluster.

Overall, K-means clustering aims to partition the dataset into k clusters such that the within-cluster variance is minimized, and data points within the same cluster are more similar to each other than to those in other clusters.[12]

3.4 Adaptive thresholding

Adaptive thresholding is a technique used in image processing to binarize an image into foreground and background regions based on local pixel intensities. Unlike global thresholding, where a single threshold value is applied to the entire image, adaptive thresholding computes threshold values for small regions of the image individually, allowing for better handling of variations in lighting and contrast across the image.It can be done by these steps:

1. Image Partitioning:

- Divide the input grayscale image I into non-overlapping regions or windows of size $n \times n$.
- For each pixel $I(x, y)$ in the image, consider the local neighborhood around it defined by the window centered at $I(x, y)$.

2. Threshold Computation:

- For each window, compute a local threshold value to binarize the pixels within that window.
- The local threshold $T(x, y)$ for each pixel $I(x, y)$ is typically computed as a function of the mean intensity (μ) of the pixel values within the window and a constant factor (c):

$$T(x, y) = \mu(x, y) - c$$

- where $\mu(x, y)$ is the mean intensity of the pixels within the window centered at $I(x, y)$.

3. Binarization:

- Binarize the image based on the computed local thresholds.
- Assign a value of 0 (background) to pixels with intensity values below the corresponding local threshold, and a value of 255 (foreground) to pixels with intensity values equal to or above the threshold.

- Mathematically, the adaptive thresholding algorithm can be summarized as follows:

$$\text{Threshold}(I(x,y)) = \begin{cases} 0, & \text{if } I(x,y) < T(x,y) \\ 255, & \text{if } I(x,y) \geq T(x,y) \end{cases}$$

- where $T(x,y) = \mu(x,y) - c$ is the local adaptive threshold computed for each pixel (x,y), and $\mu(x,y)$ represents the mean intensity of the pixels within the local window centered at (x,y).

- The parameter c controls the sensitivity of the thresholding operation. A larger value of c results in a more conservative threshold, while a smaller value allows for more variation in intensity before binarization.[13]

3.5 CNN

Employing Convolutional Neural Networks (CNNs) for litter detection capitalizes on their adeptness in extracting intricate visual features from image data. Through dataset collection, preprocessing, and model training, CNNs become proficient in identifying litter patterns. Evaluation further refines the model's performance, ensuring accurate detection. Deploying CNNs enables real-time litter identification, bolstering marine conservation efforts efficiently. CNN working methodology involves several key components:

1. Convolutional Layers:

- CNNs apply convolution operations to input images using learnable filters (also called kernels).

- These filters slide over the input image, computing dot products at each location, thereby detecting features like edges, textures, and shapes.

- Multiple filters are used in parallel to capture different features, creating feature maps for each filter.

2. Pooling Layers:

- Pooling layers downsample feature maps, reducing their spatial dimensions while retaining important information.

- Common pooling operations include max pooling and average pooling, which extract the maximum or average value within a defined region, respectively. Pooling helps to reduce computational complexity and control overfitting by providing translation invariance.

3. Activation Functions:

- Non-linear activation functions like ReLU (Rectified Linear Unit) are applied after each convolutional and pooling layer to introduce non-linearity into the network.

- ReLU sets negative values to zero, allowing the network to learn complex patterns and representations effectively.

4. Fully Connected Layers:

- After several convolutional and pooling layers, CNNs often include one or more fully connected layers.

- These layers connect every neuron from the previous layer to every neuron in the subsequent layer, enabling high-level feature learning and classification.

5. Loss Function and Optimization:

- CNNs are typically trained using supervised learning, where a loss function (e.g., cross-entropy loss for classification tasks) measures the disparity between predicted and true labels.

- Optimization algorithms like Stochastic Gradient Descent (SGD) or its variants adjust the network parameters (weights and biases) to minimize the loss function iteratively during training.

6. Backpropagation:

- Backpropagation is used to compute the gradient of the loss function with respect to the network parameters.
- The gradients are then used by the optimization algorithm to update the parameters, moving them in the direction that minimizes the loss.

7. Training and Evaluation:

- CNNs are trained on large labeled datasets, where the network learns to extract features and make predictions.
- The trained model is evaluated on a separate validation or test dataset to assess its performance and generalization ability.

Overall, CNNs excel at capturing hierarchical patterns and features in images, making them highly effective for tasks like image classification, object detection, and image segmentation.[14]

3.6 Histogram of Oriented Gradients (HOG)

HOG, or Histogram of Oriented Gradients, is a feature descriptor that is often used to extract features from image data. HOG decomposes an image into small squared cells, computes an histogram of oriented gradients in each cell, normalizes the result using a block-wise pattern, and return a descriptor for each cell. HOG, is a feature descriptor like the Canny Edge Detector, SIFT (Scale Invariant and Feature Transform). Typically, a feature descriptor converts an image of size width x height x 3 (channels) to a feature vector / array of length n. In the case of the HOG feature descriptor, the input image is of size 64 x 128 x 3 and the output feature vector is of length 3780.

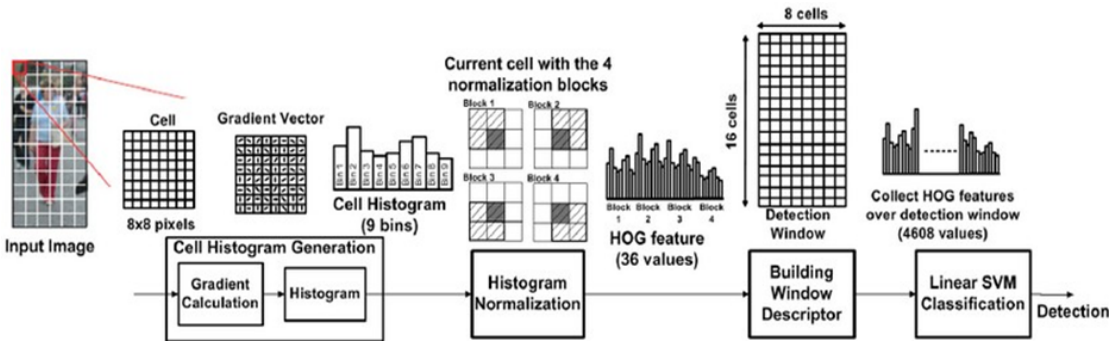


Figure 3.1: BLOCK DIAGRAM OF HOG

The steps of above block diagram are:

1. Gradient Calculation:

- Compute the gradient magnitude (M) and orientation (θ) for each pixel in the image using gradient operators like the Sobel operator:

$$G_x = \frac{\partial I}{\partial x}, \quad G_y = \frac{\partial I}{\partial y}$$

$$M = \sqrt{G_x^2 + G_y^2}, \quad \theta = \arctan 2(G_y, G_x)$$

Here, G_x and G_y represent the gradients in the x and y directions, and I is the image intensity.

2. Histogram Computation:

- Divide the image into small cells (e.g., 8×8 pixels) and accumulate gradient orientations into histograms with N orientation bins. The contribution of each pixel to the histogram is weighted by its gradient magnitude:

$$H(i) = \sum_{(p,q) \in \text{cell}} w(p,q) \times \delta(\theta(p,q) - \theta_i)$$

Here, $H(i)$ is the value of the i-th bin in the histogram, (p, q) represents the pixel location in the cell, $w(p, q)$ is the pixel intensity weight, $\delta(\cdot)$ is the Dirac delta function, and θ_i is the orientation of the i-th bin.

3. Block Normalization:

- Normalize blocks of histograms (e.g., 2×2 or 3×3 adjacent cells) to improve robustness to illumination and contrast variations. Normalization is typically done using L2-norm or L1-norm.

4. Descriptor Formation:

- Concatenate normalized block histograms to form the final feature vector. Optionally, apply further normalization or dimensionality reduction techniques such as Principal Component Analysis (PCA).

The resulting HOG descriptor represents the distribution of gradient orientations in localized regions of the image. It serves as a powerful feature for training classifiers in object detection tasks.[15]

3.7 Support Vector Machine (SVM)

SVM is a supervised learning algorithm that is capable of separating data points into different classes by finding the best hyper plane that maximizes the margin between the classes. In SVM, data points are represented as feature vectors, and the algorithm learns to classify them into different classes based on their features. SVM works by finding the hyper plane that best separates the data points into different classes while maximizing the margin, which is the distance between the hyper plane and the closest data points from each class, also known as support vectors. SVM also has some limitations, such as sensitivity to hyperparameter tuning, computational complexity for large datasets, and difficulty in handling imbalanced data.

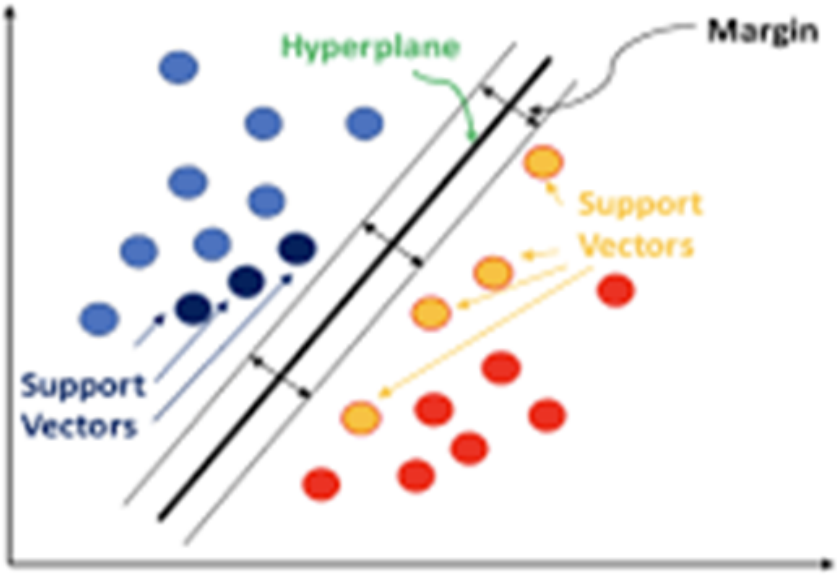


Figure 3.2: SVM Binary Classifier

Proper parameter tuning, feature engineering, and model evaluation are important for obtaining accurate and reliable results with SVM. For SVM binary classification the steps are:

1. Data Representation:

- Let's consider a dataset consisting of N samples, each with D features, and corresponding binary labels ($y_i \in \{-1, +1\}$). We represent the dataset as $(\mathbf{X}, \mathbf{y})$, where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]^T$ is the feature matrix and $\mathbf{y} = [y_1, y_2, ..., y_N]^T$ is the label vector.

2. Optimal Hyperplane:

- SVM aims to find the optimal hyperplane that separates the two classes with the maximum margin. Mathematically, this hyperplane is defined by the equation:

$$\mathbf{w}^T\mathbf{x} + b = 0$$

  where $\mathbf{w}$ is the weight vector (normal to the hyperplane) and b is the bias term.

3. Margin Maximization:

- The margin is the distance between the hyperplane and the nearest data point from each class. SVM seeks to maximize this margin.

- The margin can be calculated as:

$$\text{Margin} = \frac{2}{\|\mathbf{w}\|}$$

where $\|\mathbf{w}\|$ is the Euclidean norm of the weight vector.

4. Soft Margin Classification:

- In real-world scenarios where the data might not be perfectly separable, SVM allows for a soft margin by introducing slack variables (ξ_i) to tolerate misclassifications.

- The objective function becomes a trade-off between maximizing the margin and minimizing the classification error, often represented as:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i$$

subject to the constraints: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$, where C is the regularization parameter controlling the trade-off between margin maximization and error minimization.

5. Kernel Trick:

- SVM can efficiently handle non-linearly separable data by using a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ to implicitly map the input features into a higher-dimensional space, where a hyperplane can linearly separate the classes.

- Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid kernels.

6. Dual Optimization Problem:

- The optimization problem can be reformulated as its dual form, which involves maximizing a quadratic function subject to linear constraints.

- The solution is typically found using optimization techniques such as the Sequential Minimal Optimization (SMO) algorithm.

7. Decision Function:

- Once the optimal hyperplane is determined, the decision function for classifying new instances is given by:

$$\text{Prediction} = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

where the sign function assigns the class label based on which side of the hyperplane the data point lies.

In summary, SVM seeks to find the optimal hyperplane that maximally separates classes in feature space, with the flexibility to handle non-linearly separable data through the use of kernel functions. Its decision boundary is determined by a subset of the training data points called support vectors, making SVMs effective for binary classification tasks.[15]

3.8 YOLOV8

YOLO stands for You Only Look Once and V8 defines the version as 8. YOLO is known for its speed and accuracy in real-time object detection tasks. Each version of YOLO brings improvements and optimizations, and YOLOv8 is no exception. YOLOv8 is a powerful and versatile deep learning model designed for object detection, classification, and segmentation tasks. It is built upon the success of previous YOLO versions, offering significant improvements in terms of Speed, Accuracy, Flexibility. The model can be easily customized to different tasks and datasets, making it suitable for a wide range of applications. potential applications of yolov8 self-driving cars, robotics, surveillance, medical imaging and retail.

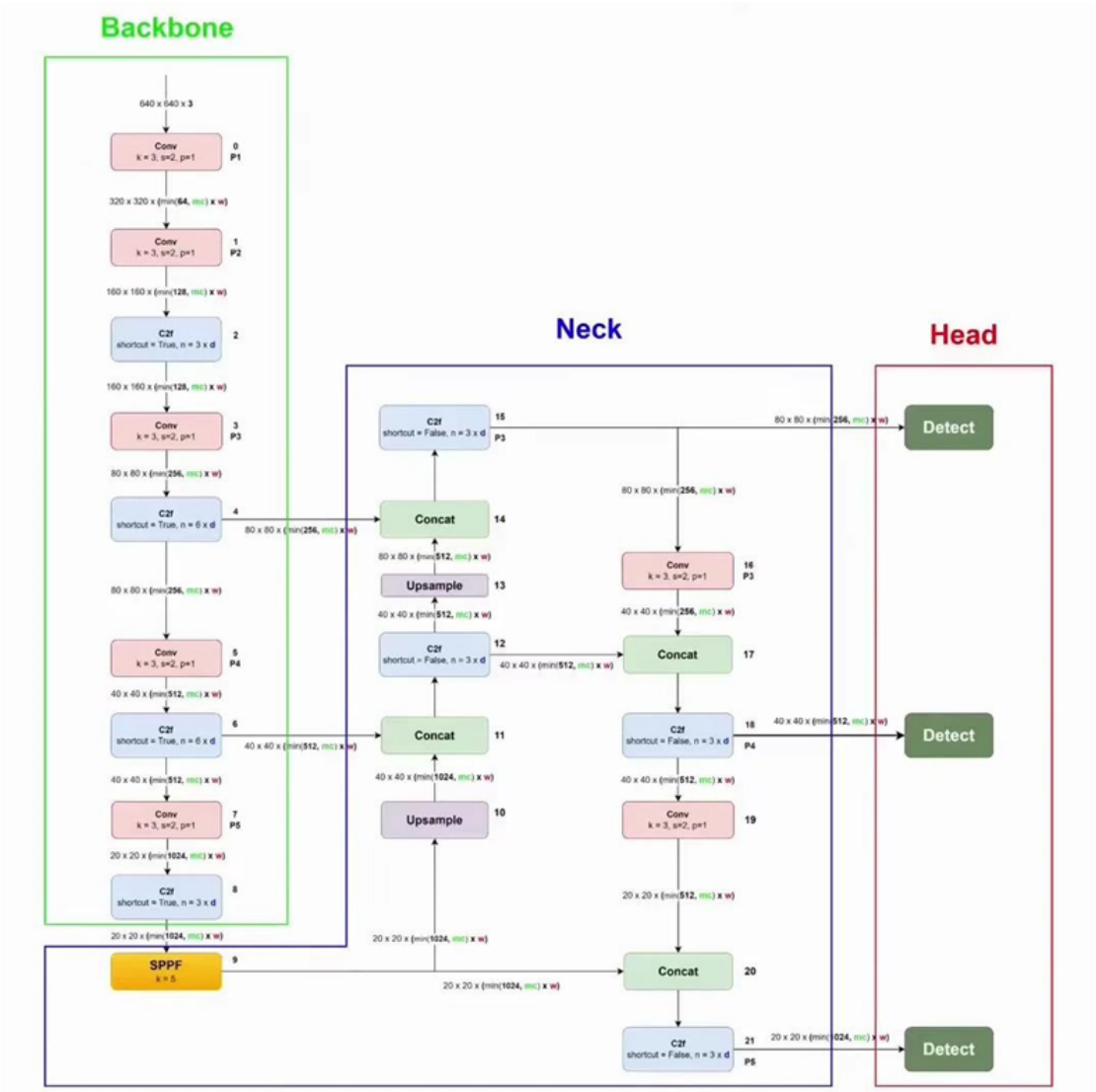


Figure 3.3: Architecture of YOLOV8

The key components of the yolov8 architecture:

1. Backbone:

- Cspdarknet53: This is the core feature extractor of yolov8, based on the Darknet architecture but with efficient cross-stage partial connections (CSP) that improve information flow and reduce computational cost.

2. Neck:

- C2F: This novel neck module replaces the traditional YOLO neck architecture. It uses an efficient design based on convolutional layers and FPN-like features to extract high-level semantic features at different scales.

3. Head:

- YOLO Head: This predicts bounding boxes, object class probabilities, and confidence scores for each detected object.

The YOLO V8 is capable of detecting 80 objects in coco data set like Person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop sign, parking meter, bench, bicycle-rack, potted plant, street sign, dog, cat, horse, sheep, cow, elephant, Bear, zebra, giraffe, backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball, gloves, skate, board, surfboard, tennis, racket, bottle, wine, glass, cup, fork, knife, spoon, bowl, banana, apple, Sandwich, orange, broccoli, carrot, hotdog, pizza, donut, cake, chair, couch, potted plant, bed, dining, table, toilet, tv, laptop, mouse, remote, keyboard, cell, phone, microwave, oven, toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear

But Objects we want to detect are

1. Cans
2. Life saver
3. Mask
4. Plastic bottle
5. Plastic cap
6. Plastic cover
7. Thermocoal
8. Wood

So to detect desired objects other than the objects present in COCO dataset we are going to use Transfer Learning[16]

3.8.1 Transfer Learning

Transfer learning is a popular technique in deep learning, and it can be applied to YOLOv8 (You Only Look Once version 8) to enhance its performance on a specific task, even when limited labeled data is available. Transfer learning involves using a pre-trained model on a large dataset and fine-tuning it for a target task. In the context

of YOLOv8, you can leverage a pre-trained model on a large dataset like COCO and fine-tune it for your custom object detection task.

Here are the general steps for transfer learning with YOLOv8:

1. Install YOLOv8
2. Prepare Custom Dataset
3. Download Pre-trained Weights
4. Configure YOLOv8 for Transfer Learning
5. Train YOLOv8
6. Evaluate and Fine-tune
7. Export Trained Weights
8. Inference[16]

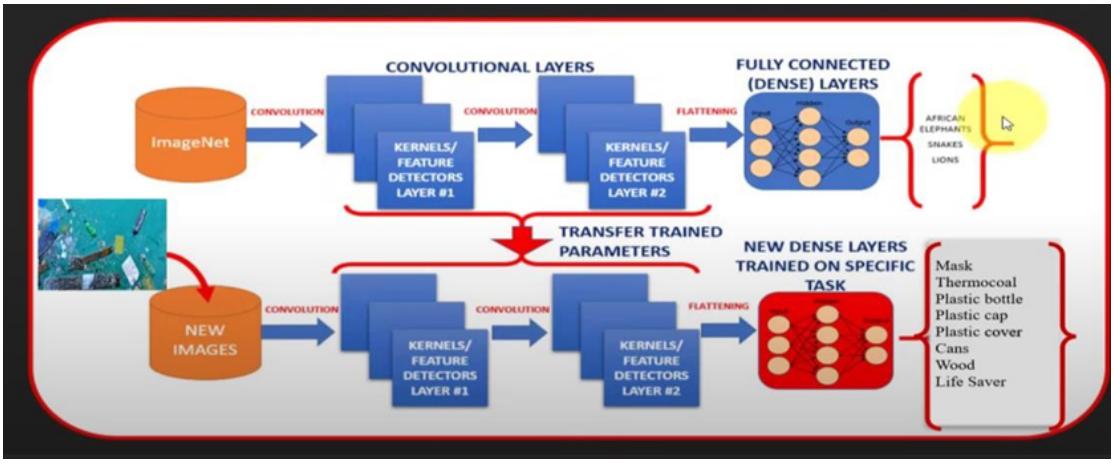


Figure 3.4: Transfer Learning Model

Chapter 4

EDGE COMPUTING

For real time application we are implementing the model on Raspberry Pi 4. It takes live input using camera module.

4.1 Setting of Raspberry Pi 4

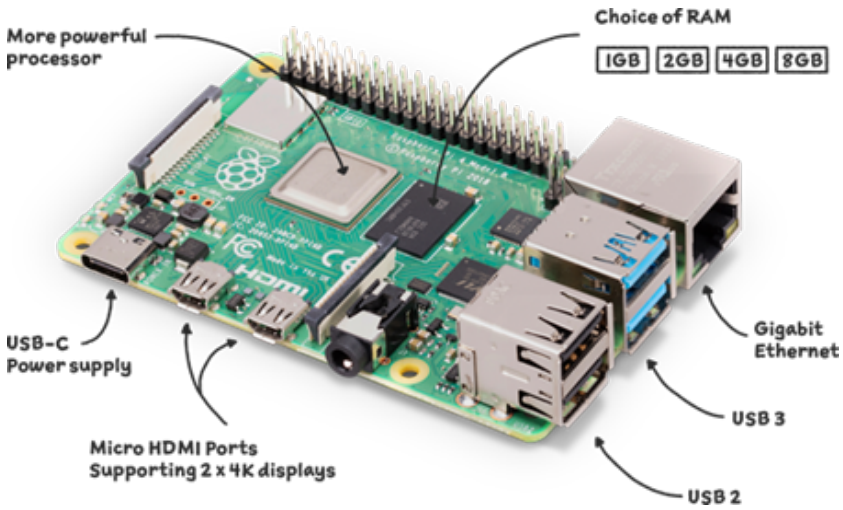


Figure 4.1: Raspberry Pi 4

The Raspberry Pi is a series of single-board computers developed by the Raspberry Pi Foundation, a UK charity focused on promoting computing education. It is a versatile and affordable computer that has gained popularity in various fields, from education to industrial applications.

Purpose: The Raspberry Pi aims to make computing accessible and affordable for everyone, from beginners to experts.

Features: It offers 2-3 times the speed of previous generations and includes GPIO

(general-purpose input/output) pins for controlling electronic components, making it suitable for physical computing and IoT projects.

Models: There have been several generations of Raspberry Pi models, from Pi 1 to 4, including variants like Model A and Model B. The latest models include the Raspberry Pi 4 and Pi Zero 2.

Applications: People use the Raspberry Pi for learning programming skills, building hardware projects, home automation, implementing Kubernetes clusters, edge computing, and even industrial applications.

The Raspberry Pi 4 supports multiple operating systems, including both Linux-based and Android-based distributions. We used Raspberry Pi OS (previously Raspbian) for our project. It is a Official OS for Raspberry Pi, offering stability and performance optimized for Raspberry Pi hardware.

Raspberry Pi 4 Hardware Components:

1. System-on-a-Chip:

- Broadcom BCM2711

- Quad-core Cortex-A72 @ 1.5 GHz (B0 Revision) / 1.8 GHz (C0 Revision)

2. Memory:

- LPDDR4 @ 3200 MHz

- Available in 1GB, 2GB, 4GB, or 8GB configurations

3. Connectivity:

- 802.11ac Wi-Fi / Bluetooth 5.0

- Gigabit Ethernet

4. Video and Sound:

- 2 x micro-HDMI ports supporting 4K@60Hz displays via HDMI 2.0

- MIPI DSI display port 10

- MIPI CSI camera port

- 4-pole stereo output and composite video port

5. Ports:

- 2 x USB 3.0 ports

- 2 x USB 2.0 ports

- Raspberry Pi standard 40-pin GPIO header

6. Graphics:

- VideoCore VI @ 500 MHz

- H.265 (4kp60 decode), H264 (1080p60 decode, 1080p30 encode)

- OpenGL ES 3.1, Vulkan 1.0

7. Power:

- Micro-SD card slot for loading operating system and data storage

- Power input: 5V @ 3A via USB-C connector or GPIO header[17]

USB-C:

The Raspberry Pi 4 features a USB-C port for power input. However, unlike typical USB-C specifications, the Raspberry Pi 4 accepts only 5V DC input through this port, regardless of whether the connected power source supports higher voltages such as USB Power Delivery (PD). The Raspberry Pi 4's USB-C power circuitry includes a single resistor, which causes it to ignore the USB-C PD signaling and always expect a 5V input.

4.1.1 Camera Module

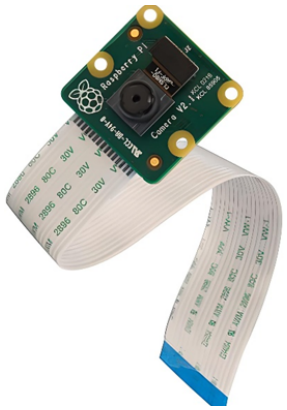


Figure 4.2: Camera Module

We used Raspberry Pi Camera Module Rev 1.3, also known as the Raspberry Pi Camera Module V1.3, It has the following specifications:

1. Image Sensor: OmniVision OV5647 CMOS image sensor

2. Resolution: 5 megapixels (2592 × 1944 pixels)

3. Pixel Size: 1.4 m x 1.4 m

4. Lens: Fixed focus lens

5. Aperture: f/2.9

6. Field of View (FOV): 54° horizontal, 41° vertical, 64° diagonal

7. Supported Video Modes:

- 1080p at 30 frames per second (fps)

- 720p at 60 fps

- 640 × 480p at 60/90 fps

4.1.2 PuTTY

PuTTY is a versatile and widely used terminal emulator and network connection manager primarily designed for Windows. It allows users to securely connect to remote systems, perform system administration tasks, transfer files, and troubleshoot network issues

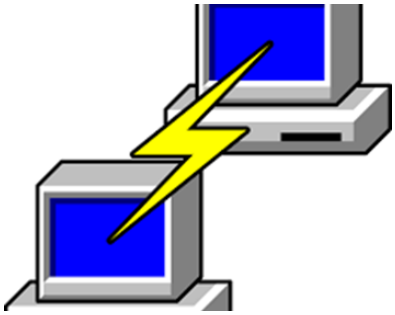


Figure 4.3: Putty

Key Features of PuTTY:

SSH Support: Provides robust support for Secure Shell (SSH) protocol,ensuring encrypted connections to remote servers.

Terminal Emulation: Offers a powerful terminal emulation environment for managing servers and networking devices.

Session Management: Allows users to save connection configurations as ”sessions” for easy access to frequently used servers.

Customization: Users can tailor settings to their preferences, adjusting font size, colors,

keyboard shortcuts, and more.

By using command “ifconfig” we obtained ip address of the raspberry pi 4

```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 169.254.154.39  netmask 255.255.0.0  broadcast 169.254.255.255
        inet6 fe80::e65f:1ff:fe95:4824  prefixlen 64  scopeid 0x20<link>
        ether e4:5f:01:95:48:24  txqueuelen 1000  (Ethernet)
        RX packets 205  bytes 18292 (17.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 83  bytes 14105 (13.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 19  bytes 2245 (2.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19  bytes 2245 (2.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.137.59  netmask 255.255.255.0  broadcast 192.168.137.255
        inet6 fe80::7312:58e9:6cf5:4def  prefixlen 64  scopeid 0x20<link>
        ether e4:5f:01:95:48:25  txqueuelen 1000  (Ethernet)
        RX packets 554  bytes 773567 (755.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 553  bytes 41994 (41.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 4.4: Putty Terminal

4.1.3 VNC Viewer

VNC Viewer is a remote desktop software that allows users to access and control computers over a network. It provides instant remote access to Mac, Windows, and Linux computers from anywhere in the world. Here are some key points about VNC Viewer based on the search results :

Figure 4.5: VNC viewer

Features: VNC Viewer allows users to view and control their computer’s desktop, as well as interact with the remote screen as if they were physically present in front of it.

Compatibility: VNC Viewer supports various platforms, including Windows, Mac, iOS, and Android, enabling users to connect to their computers from different devices.

Security: All sessions in VNC Viewer are encrypted end-to-end, ensuring secure remote access to computers.

Usage: We need to download RealVNC Connect on each computer they want to control and sign in to VNC Viewer on their device using RealVNC account credentials to establish connections.

VNC Viewer simplifies remote access and control of computers, making it a valuable tool for individuals who need to manage their systems from a distance.

In VNC viewer we accessed raspberry pi 4 using IP address.

4.1.4 GPS Module

A GPS module consists of a receiver chip and antenna, receiving signals from satellites to determine precise location. It may include a microcontroller for data processing. Communication interfaces like UART enable connection to external devices. They operate on various voltages, typically 3.3V or 5V. GPS modules are utilized in navigation for vehicles, boats, and aircraft, facilitating route planning and real-time position tracking. They aid mapping, surveying, and geographic information systems, offering accurate location data. GPS modules synchronize time in telecommunications and scientific research, referencing atomic clocks on satellites. Geotagging photos and videos is another common application, associating media with geographic coordinates. With versatile capabilities, GPS modules play crucial roles in navigation, tracking, mapping, timing, and geotagging, supporting diverse technological advancements and systems.

Figure 4.6: GPS Module

4.2 Litter detection on Raspberry Pi 4

As a drone traverses the ocean, its onboard camera records video footage, which undergoes immediate processing by a Raspberry Pi deployed with Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM) algorithms. Upon detecting litter within the video stream, the Raspberry Pi interfaced with a GPS module precis determine the location coordinates. Subsequently, this location data is transmitted to an autonomous vehicle poised for cleanup operations. Through this seamless integration of drone-based litter detection, GPS localization, and autonomous vehicle navigation, the system facilitates swift and targeted cleanup actions, mitigating the detrimental impact of oceanic pollution.

4.3 Litter recognition on Raspberry Pi 4

The YOLOv8 model, pre-trained for object recognition, is deployed on a Raspberry Pi 4, with the custom model weight file of approximately 23 MB of storage. This implementation significantly enhances the efficiency of the cleaning process by enabling real-time object recognition. As a result, the system can swiftly identify and classify objects of interest, streamlining the cleanup efforts. Leveraging the compact and powerful computing capabilities of the Raspberry Pi 4, the deployment of the YOLOv8 model facilitates accurate and rapid detection of objects, optimizing the overall cleaning process.

Chapter 5

EVALUATION PARAMETERS

5.1 Parameters

Table 5.1: Evaluated Parameters

Parameters	Value
Inference Time	110ms
Recall	0.77
Precision	0.73

5.2 Metrics Graphs

Table 5.2: Confusion Matrix

	can	**lifesaver**	**mask**	**bottle**	**cap**	**cover**	**thermocol**	**wood**	**background**
can	0.93			0.02					0.01
lifesaver		0.86							0.01
mask			0.86			0.01			0.04
bottle	0.02			0.76	0.02	0.04	0.00	0.04	0.48
cap					0.60				0.23
cover						0.60			0.09
thermocol							0.81	0.00	0.08
wood								0.64	0.06
background	0.05	0.14	0.14	0.22	0.38	0.35	0.19	0.32	

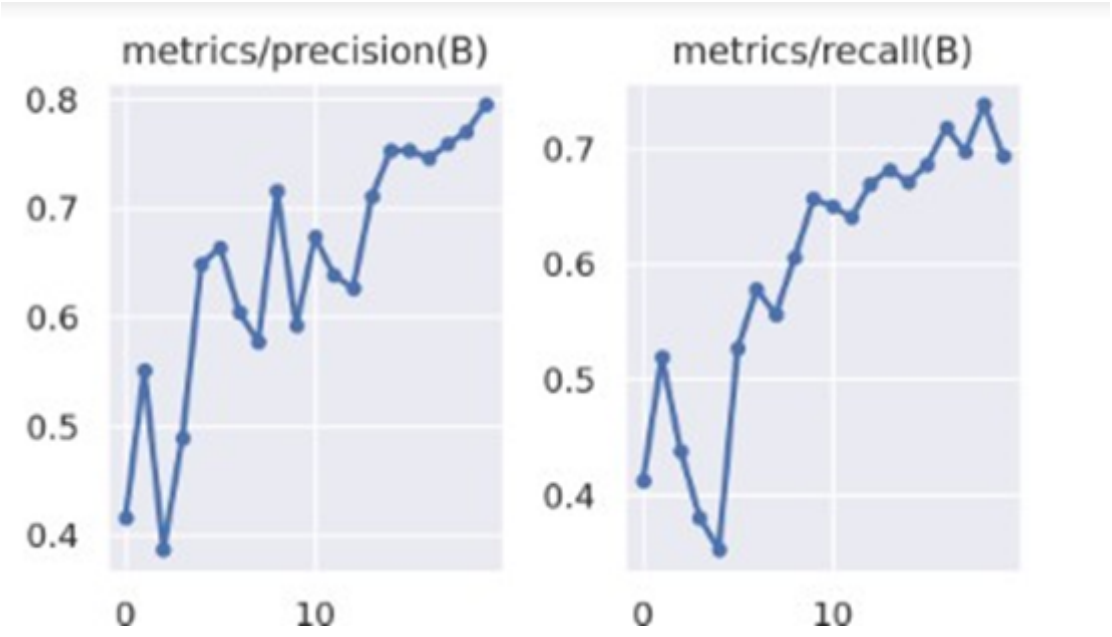


Figure 5.1: Precision and Recall graphs

Figure 5.2: Training loss and Validation loss graphs

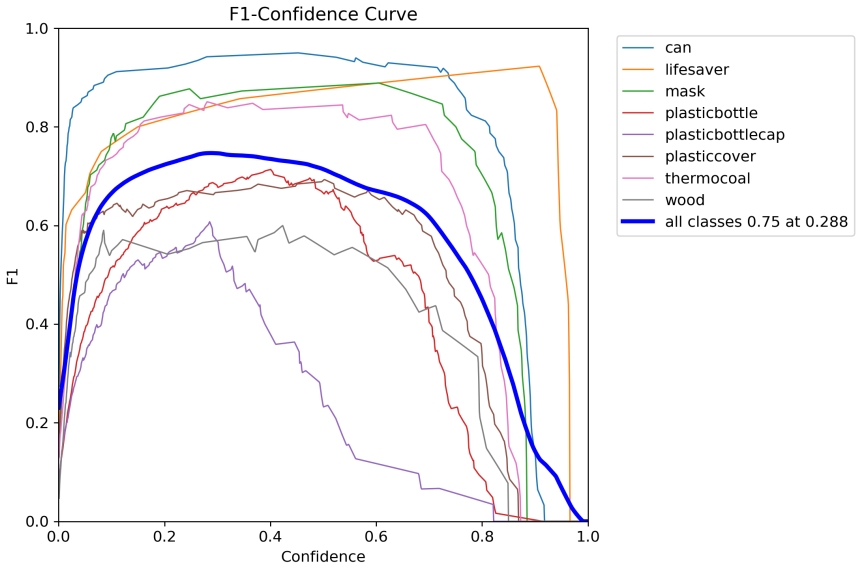


Figure 5.3: F1 curve

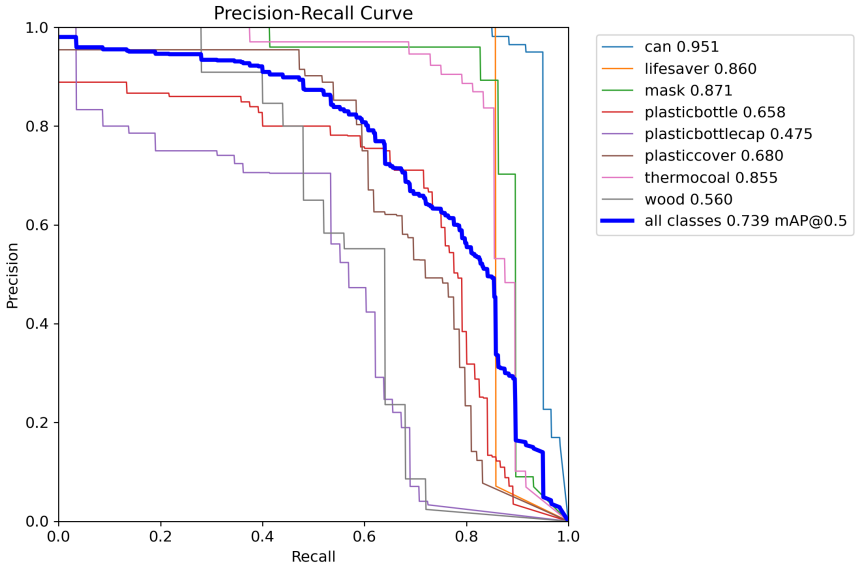


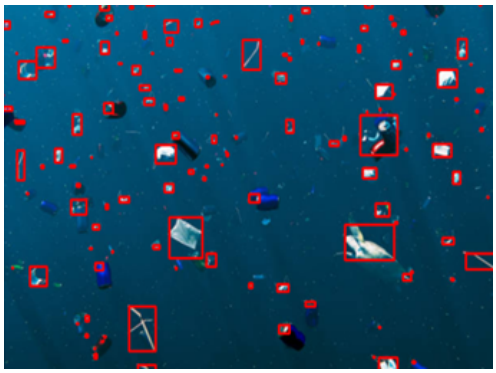
Figure 5.4: PR curve

Chapter 6

RESULTS AND DISCUSSION

6.1 Results of image processing techniques

(a) Detection By Canny Edge



(b) Detection By K-means Clustering

Figure 6.2: Detection By Adaptive thresolding

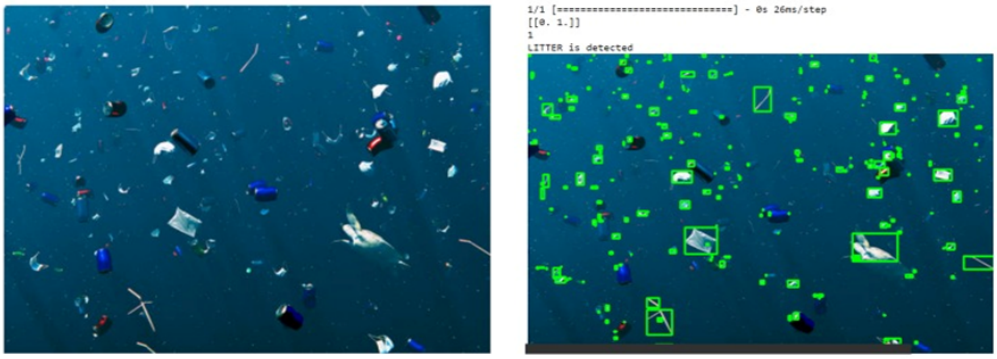


Figure 6.3: Detection By CNN

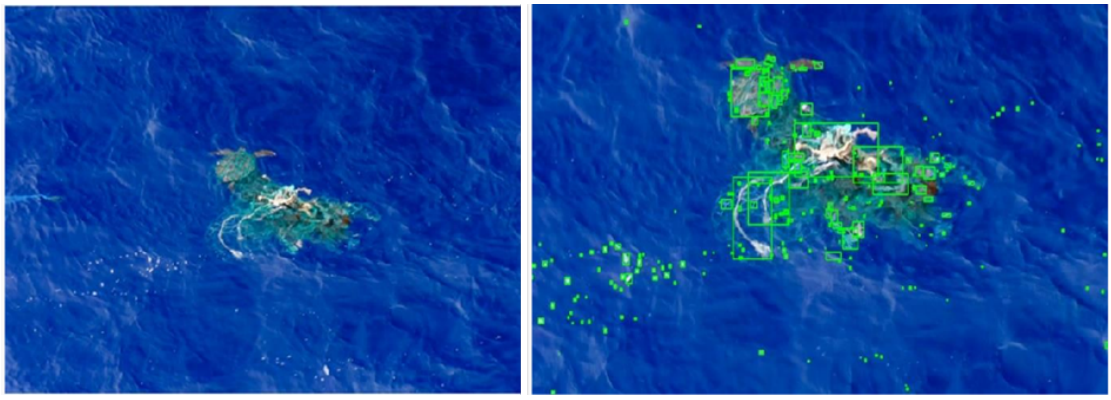


Figure 6.4: Detection by HOG and SVM

Figure 6.5: Result by YOLOV8

Figure 6.6: Detected Litter by HOG and SVM on Raspberry Pi

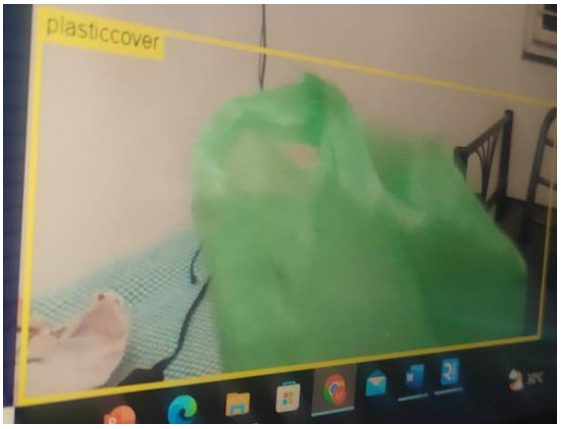


Figure 6.7: Recognition of Litter by YOLO V8 on Raspberry Pi

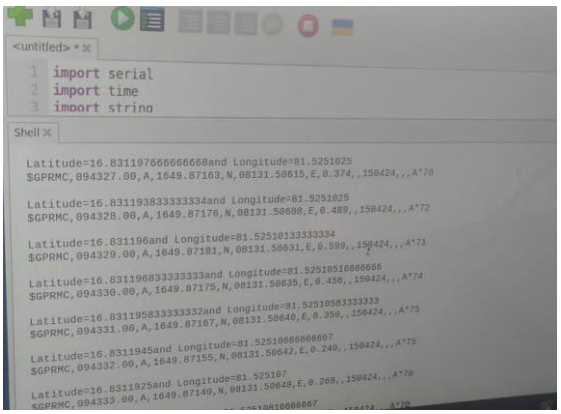


Figure 6.8: Location of detected litter

Chapter 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

In this thesis, we conducted a thorough evaluation of various computer vision techniques and machine learning algorithms to tackle the challenge of marine litter detection and localization. Despite shortcomings observed in methods such as Canny edge detection, adaptive thresholding, k-means clustering, and convolutional neural networks regarding accuracy, the integration of Histogram of Oriented Gradients (HOG) with Support Vector Machine (SVM) proved highly effective in discerning marine debris. Notably, the augmentation of our approach with YOLOv8 object recognition, leveraging transfer learning for custom object detection, significantly enhanced object classification capabilities.

By implementing these advanced techniques on a Raspberry Pi 4 platform and integrating GPS modules for precise geographic coordinate acquisition, we successfully achieved real-time detection and localization of marine litter.The findings of this study underscore the transformative potential of leveraging advanced computer vision and machine learning technologies to address the urgent issue of marine pollution. By combining cutting-edge methodologies with transfer learning-enabled custom object detection, we have taken significant strides toward the preservation of marine ecosystems.This research contributes to a growing body of knowledge aimed at fostering sustainable practices and safeguarding our planet's natural resources for future generations.

7.2 Future Scope

The future scope of this thesis project involves implementing a comprehensive and multi-dimensional strategy to improve the efficiency and effectiveness of marine litter detection and cleanup initiatives. Key objectives include expanding the dataset size to enhance the robustness of machine learning models, deploying algorithms directly onto litter-collecting robots to enable real-time decision-making, and developing motion detection and tracking techniques to address the assumption of stationary litter. Additionally, the project aims to integrate predictive modeling to anticipate the trajectory of moving litter objects and explore alternative communication methods for long-distance data transmission in areas with limited Wi-Fi connectivity. Through these endeavors, the thesis endeavors to offer a robust and impactful solution for mitigating marine litter pollution and fostering sustainable environmental stewardship. This will be achieved by continuously refining technological capabilities, broadening the scope of applications.

Bibliography

[1] Michael Fulton, Jungseok Hong, Md Jahidul Islam, and Junaed Sattar. Robotic detection of marine litter using deep visual detection models. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5752–5758, 2019.

[2] Srikanta Sannigrahi, Bidroha Basu, Arunima Sarkar Basu, and Francesco Pilla. Detection of marine floating plastic using sentinel-2 imagery and machine learning models. *arXiv preprint arXiv:2106.03694*, 2021.

[3] Marian-Daniel Iordache, Liesbeth De Keukelaere, Robrecht Moelans, Lisa Landuyt, Mehrdad Moshtaghi, Paolo Corradi, and Els Knaeps. Targeting plastics: Machine learning applied to litter detection in aerial multispectral images. *Remote Sensing*, 14(22):5820, 2022.

[4] Muhammad Waqas, Man Sing Wong, Alessandro Stocchino, Sawaid Abbas, Sidrah Hafeez, and Rui Zhu. Marine plastic pollution detection and identification by using remote sensing-meta analysis. *Marine Pollution Bulletin*, 197:115746, 2023.

[5] Luca Fallati, Annalisa Polidori, Christian Salvatore, Luca Saponari, Alessandra Savini, and P Galli. Anthropogenic marine debris assessment with unmanned aerial vehicle imagery and deep learning: A case study along the beaches of the republic of maldives. *Science of The Total Environment*, 693:133581, 2019.

[6] Odei Garcia-Garin, Toni Monle n Getino, Pere L pez Brosa, Asunci n Borrell, Alex Aguilar, Ricardo Borja-Robalino, Luis Cardona, and Morgana Vighi. Automatic detection and quantification of floating marine macro-litter in aerial images:

Introducing a novel deep learning approach connected to a web application in r. *Environmental Pollution*, 273:116490, 2021.

[7] Brendan Chongzhi Corrigan, Zhi Yung Tay, and Dimitrios Konovessis. Real-time instance segmentation for detection of underwater litter as a plastic source. *Journal of Marine Science and Engineering*, 11(8):1532, 2023.

[8] Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi. Real-time flying object detection with yolov8. *arXiv preprint arXiv:2305.09972*, 2023.

[9] Bhanuka Gamage, Thanh-Toan Do, Nicholas Seow Chiang Price, Arthur Lowery, and Kim Marriott. What do blind and low-vision people really want from assistive smart devices? comparison of the literature with a focus study. In *Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–21, 2023.

[10] Cheng Siong Chin, Aloysius Bo Hui Neo, and Simon See. Visual marine debris detection using yolov5s for autonomous underwater vehicle. In *2022 IEEE/ACIS 22nd International Conference on Computer and Information Science (ICIS)*, pages 20–24. IEEE, 2022.

[11] P. Durgadevi, T. Akilan, Ajitesh Pradhan, A.S. Mohammed Shariff, Neha Yadav, and Pranav Uppal. Canny edge detection techniques for image segmentation. In *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 986–989, 2022.

[12] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[13] Payel Roy, Saurab Dutta, Nilanjan Dey, Goutami Dey, Sayan Chakraborty, and Ruben Ray. Adaptive thresholding: A comparative study. In *2014 Interna-*

tional Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pages 1182–1186, 2014.

[14] Keiron O'shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[15] Ngo-Doanh Nguyen, Duy-Hieu Bui, and Xuan-Tu Tran. A novel hardware architecture for human detection using hog-svm co-optimization. In *2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 33–36, 2019.

[16] Misha Urooj Khan, Mahnoor Dil, Maham Misbah, Farooq Alam Orakazi, Muhammad Zeshan Alam, and Zeeshan Kaleem. Translearn-yolox: Improved-yolo with transfer learning for fast and accurate multiclass uav detection. In *2023 International Conference on Communication, Computing and Digital Systems (C-CODE)*, pages 1–7, 2023.

[17] V Viswanatha, RK Chandana, and AC Ramachandra. Iot based smart mirror using raspberry pi 4 and yolo algorithm: A novel framework for interactive display. *Indian Journal of Science and Technology*, 15(39):2011–2020, 2022.

Appendices

Appendix A

Code Attachments

A.1 HOG and SVM

The below code snippet demonstrates a data preprocessing pipeline for marine litter detection using HOG features and SVM. It traverses through a directory structure containing labeled image data, extracting HOG features from each image after resizing and converting to grayscale. The extracted features are stored alongside corresponding class labels. HOG parameters such as orientations, pixels per cell, and cells per block are defined for feature extraction. The code emphasizes modularity and scalability, facilitating easy integration with an SVM classifier for subsequent training and evaluation.

```
data_path='/content/train'
# Define the parameters for HOG feature extraction
orientations = 9
pixels_per_cell = (8, 8)
cells_per_block = (3, 3)
visualize = False
normalize = True

# Initialize empty lists to store the features and labels
hog_features = []
labels = []

# Loop through each subdirectory in the dataset
for class_folder in os.listdir(data_path):
    class_path = os.path.join(data_path, class_folder)
    if not os.path.isdir(class_path):
        continue
    # Loop through each image file in the subdirectory
    for file_name in os.listdir(class_path):
        image_path = os.path.join(class_path, file_name)
        if not os.path.isfile(image_path):
            continue
        # Load the image and convert it to grayscale
        image = io.imread(image_path, as_gray=True)
        image = cv2.resize(image,(128,128))
```

```
        # Extract the HOG features from the image
        hog_feature = feature.hog(image, orientations=orientations,
                                  pixels_per_cell=pixels_per_cell,
                                  cells_per_block=cells_per_block,
                                  visualize=visualize)#, normalize=
    normalize)
     # Append the features and label to the corresponding lists
         hog_features.append(hog_feature)
         print(hog_features)
         labels.append(class_folder)
# Convert the lists to numpy arrays
hog_features = np.array(hog_features)
labels1 = np.array(labels)
#SVM CODE
from sklearn import svm
clf = svm.SVC(kernel='linear')
clf.fit(hog_features, labels1)
```

A.2 GPS Module

The below code snippet establishes a serial connection with a GPS device and continuously reads NMEA data from it. It utilizes the ‘serial‘ library to communicate with the GPS module through a serial port, parsing the incoming NMEA sentences using ‘pynmea2‘ library. Specifically, it focuses on extracting latitude and longitude information from the GPRMC (Recommended Minimum Specific GNSS Data) sentences. Once parsed, it prints the raw NMEA data and the extracted latitude and longitude values. This script can serve as a foundational component for GPS-based applications such as vehicle tracking, navigation systems, or geotagging.

```
import serial
import time
import string
import pynmea2
while True:
    ser=serial.Serial("/dev/ttyAMA0", baudrate=9600, timeout=1)
    dataout =pynmea2.NMEAStreamReader()
    newdata=ser.readline()
    #print(newdata)
    if '$GPRMC' in str(newdata):
        print(newdata.decode('utf-8'))
        newmsg=pynmea2.parse(newdata.decode('utf-8'))
        lat=newmsg.latitude
        lng=newmsg.longitude
        gps = "Latitude=" + str(lat) + "and Longitude=" +str(lng)
        print(gps)
```