VEHICLE DETECTION USING SVM: SOFT COMPUTING

PROJECT REPORT

(Project Semester January-April 2025)

(Object Detection Using SVM)

Submitted by

(Alfeen K Afsal)

Registration No: 12320872

Programme and Section: B. Tech CSE, K23WT

Course Code: INT256

Under the Guidance of

(Sheveta, UID: 16856)

Discipline of CSE/IT

Lovely School of Computer Science and Engineering

Lovely Professional University, Phagwara

CERTIFICATE

This is to certify that Alfeen K Afsal bearing Registration no. 12320872 has completed INT256 project titled, "**Vehicle Detection using SVM**" under my guidance and supervision. To the best of my knowledge, the present work is the result of his/her original development, effort and study.

Signature and Name of the Supervisor

Designation of the Supervisor

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab.

Date: 20th April, 2025

DECLARATION

I, Savant Kumar Jena, student of B. Tech under CSE/IT Discipline at, Lovely Professional University, Punjab, hereby declare that all the information furnished in this project report is based on my own intensive work and is genuine.

Date: 20th April, 2025 Signature: 

Registration No. 12320872 Name of the student: Alfeen K Afsal

TABLE OF CONTENTS

ACKNOWLEDGEMENT

The completion of the **"Vehicle/Object Detection System"** report would not have been possible without the invaluable support, guidance, and encouragement from various individuals and institutions. I am deeply grateful to my college institution, **Lovely Professional University (LPU)**, for providing me with an outstanding academic environment, access to cutting-edge tools, and opportunities to explore and innovate in the domain of computer vision and machine learning. The university's strong emphasis on hands-on learning and research has played a key role in the development of this project.

I would like to express my sincere gratitude to my mentor, **Sheveta**, for their continuous guidance, motivation, and technical insights throughout this journey. Their expertise in machine learning techniques, particularly in **feature extraction, model training, and evaluation**, was instrumental in helping me understand the application of **HOG descriptors, SVM classifiers**, and **YOLO frameworks** in real-world object detection tasks. Their valuable feedback and encouragement helped me overcome challenges and ensure the successful completion of the project

I am also immensely thankful to my colleagues, for their collaboration, brainstorming sessions, and technical assistance throughout this endeavor. Their diverse perspectives and willingness to share knowledge enriched the project and made the process both rewarding and enjoyable.

This project stands as a testament to the collective effort and support I have received, and I am truly honored to have worked with such a dedicated team and institution. Thank you all for your contributions to this journey.

1. INTRODUCTION:

Object detection using Support Vector Machine (SVM) is a specialized application within the broader field of computer vision that combines object classification and localization tasks. While image classification assigns a class label to an entire image, and object localization identifies the position of objects within an image using bounding boxes, object detection brings both tasks together—locating and classifying multiple objects within the same frame. When practitioners refer to object recognition, they typically mean this combined process of detecting and identifying objects.

In an SVM-based object detection system, the algorithm is trained to differentiate between object classes by drawing optimal decision boundaries in a high-dimensional feature space. Feature extraction techniques such as Histogram of Oriented Gradients (HOG), Scale-Invariant Feature Transform (SIFT), or Local Binary Patterns (LBP) are used to transform raw image data into numerical features suitable for SVM classification. The trained SVM model then predicts object classes and helps in identifying the bounding boxes using sliding windows and region proposal techniques.

This method is widely applied across various domains, from surveillance and facial recognition to industrial automation and intelligent traffic systems. For instance, in security applications, SVM-based object detection can identify and track authorized personnel in restricted areas. In e-commerce, it enhances visual search engines, helping users find similar products based on image content. These capabilities highlight the practical importance of SVM in real-time object detection systems, where accuracy, speed, and interpretability are essential

2. SOURCE OF DATASET:

Link: [Binary-Prediction-with-a-Rainfall-Dataset-DL-AI/train.csv at main · SSJ0406/Binary-Prediction-with-a-Rainfall-Dataset-DL-AI]

This repository is publicly available and has a comprehensive set of rainfall data collected from various areas in India over an extensive time span. The dataset includes major parameters

3. EDA (EXPLORATORY DATA ANALYSIS) PROCESS:

Exploratory Data Analysis (EDA) played a crucial role in the development of the Object Detection System using Support Vector Machine (SVM), serving as the foundation for understanding the structure and characteristics of the image dataset. Through a combination of visualizations, statistical summaries, and feature distribution analysis, EDA helped uncover meaningful patterns, class imbalances, noise, and potential correlations within the extracted features—such as Histogram of Oriented Gradients (HOG), color histograms, and edge orientations. This process provided valuable insights that guided the selection of relevant features, preprocessing techniques like normalization and dimensionality reduction (e.g., PCA), and the configuration of the SVM classifier. The comprehensive EDA ensured the model was trained on a clean, informative, and well-structured dataset, ultimately enhancing classification accuracy and detection performance. The following outlines the detailed EDA steps undertaken during the system's development.:

- **Dataset Loading and Initial Exploration**:
    - The dataset was imported from the dataset saved as a CSV file using the pandas library, creating a DataFrame for analysis.
    - The dataset's dimensions were inspected with data.shape, providing the total number of rows and columns, which helped gauge the dataset's scale and complexity.
    - The first and last five rows were examined using data.head() and data.tail(), respectively, to understand the data's structure, feature types, and sample values.
    - Column names were standardized using data.columns.str.strip() to eliminate any whitespace, ensuring consistency in referencing features during analysis and modeling.
    - The data.info() method was invoked twice to summarize data types (e.g., float64 for numerical features, object for the target variable) and non-null counts, offering an early indication of potential missing values.
    - This initial inspection confirmed the presence of important features.
- **Missing Value Assessment and Handling**:

- Missing values were quantified using data.isnull().sum(), which revealed null entries in the columns.
- To address these gaps, mean imputation was applied: data['column'].fillna(data['column'].mean(), inplace=True) and similarly for other gaps.
- Mean imputation was chosen as a simple yet effective strategy to preserve data volume, especially given the numerical nature of these features and the assumption that their distributions were not heavily skewed.
- Post-imputation, data.isnull().sum() was re-run to confirm that no missing values remained, ensuring a complete dataset for analysis and modeling.

- **Duplicate and Unique Value Verification**:
  - The dataset was checked for duplicate rows using data.duplicated() to identify any redundant entries that could bias the model or inflate performance metrics.
  - No duplicates were explicitly removed, suggesting the dataset was clean in this regard or that duplicates were considered negligible.
  - The target variable was analyzed with data.unique(). This step validated the dataset's suitability.

Target Variable Distribution Analysis:

- A class distribution bar plot was created to visualize the frequency of different object categories (e.g., person, car, bicycle) using `sns.countplot(x="label", data=annotations)` along with `plt.figure(figsize=(6,4))`.
- This visualization revealed how often each object class appeared in the dataset, helping determine if the dataset was balanced or skewed toward certain object types.
- The simplicity of the plot made it highly effective for quickly assessing whether class imbalance was present, which is crucial for object detection models like YOLO and SVM-HOG.
- Class imbalance can lead to biased detection, where the model may detect more frequent objects (like "person") more accurately than rarer ones (like "bicycle" or "traffic light").
- Based on the class distribution, further data augmentation or synthetic sampling was considered to ensure more balanced learning across all object categories, especially for training the SVM classifier and fine-tuning YOLO for better generalization.
- **Key Insights and Implications**:

- A class distribution bar plot was created using `sns.countplot(x="label", data=annotations)` with `plt.figure(figsize=(6,4))` to analyze the frequency of various object categories (such as person, car, and bicycle) within the annotated dataset. This visualization played a crucial role in assessing class imbalance, which is particularly important for object detection models like YOLO and SVM with HOG features.
- The bar plot provided a quick and intuitive overview of how frequently each object type appeared in the dataset. A disproportionate representation of certain classes (e.g., an excess of "person" objects compared to "bicycle" or "traffic light") can lead to biased detection performance, where the model tends to perform well on dominant classes but struggles with underrepresented ones.
- To address this, strategies such as data augmentation, oversampling of rare object classes, and fine-tuning anchor boxes in YOLO were considered. These approaches aim to create a more balanced learning environment, enhancing the detection accuracy and generalization capability of the model across all object categories.

---

4. Data Preprocessing

Data preprocessing is a crucial phase in preparing the dataset for object detection tasks using machine learning algorithms such as SVM, HOG, and YOLO. This phase aims to address various issues identified in the Exploratory Data Analysis (EDA) and transforms the data into a format suitable for training these models. The following details the structured preprocessing pipeline implemented for object detection:

- **Feature Extraction Using HOG:**
  - The raw image data was processed using Histogram of Oriented Gradients (HOG) to extract useful features that represent the shape and structure of objects in the images.
  - The hog_calculator() function was used to convert each image into HOG descriptors, which capture gradient intensity and orientation across the image.
  - This transformation was essential for converting visual information into numerical features that the SVM classifier can process effectively.
- **Resizing and Normalization:**
  - Images were resized to a consistent size using the load_images() function, which helps standardize input to the model. Consistent image sizes ensure that the object detection

model doesn't face issues with scale variations across different inputs.

- Pixel values were normalized to a range of 0-1 to speed up training and help the model converge faster during the training phase.

- **Bounding Box Encoding:**
  - For object detection tasks, the target variable is often the **bounding box coordinates** around the object. These were encoded into the dataset in the format [x_min, y_min, x_max, y_max], representing the top-left and bottom-right corners of the box.
  - For YOLO, the bounding box was normalized based on the image dimensions to ensure consistency across different images, following the YOLO format of [x_center, y_center, width, height].
- **Data Augmentation:**
  - To improve the model's generalization ability, data augmentation techniques such as random flips, rotations, and zooms were applied to the training images. This helped simulate a more diverse dataset and reduce the risk of overfitting.
  - This step was essential for object detection as it increases the variety of training examples, making the model more robust to variations in object appearance and position.
- **Class Label Encoding:**
  - For YOLO, the class labels for different objects (e.g., vehicle, person, animal) were encoded into numerical values, which the model used to categorize the objects during training.
  - Similarly, for the SVM classifier, the object classes were mapped to numerical labels (e.g., 0 for non-vehicle, 1 for vehicle) to facilitate classification during the detection phase

Feature and Target Separation:

- The dataset was split into features (X) and target (y). The features consisted of the image data and object-specific attributes (e.g., pixel values, HOG descriptors), while the target variable was the bounding box coordinates and object class labels.
- The image data (X) contained the pixel values or extracted HOG features for each image, and the target variable (y) included the bounding box coordinates (for object location) and class labels (for object categorization).
  - This separation isolated the input data (e.g., HOG features for SVM or pixel values for YOLO) from the target data, making the dataset

suitable for training the object detection models.

5. EVALUATION METRICS:

1)model_exists()

Role:

- This function checks if a pre-trained model exists to avoid retraining the model every time the program runs. If the model already exists, it loads the model, saving time and computational resources.

Explanation:
If a pre-trained model file is found, it skips the training process. Otherwise, it triggers the training function to create a new model.

Mathematical Concept:
The concept here is to check for the existence of a model, which is essentially a file-check operation.

Formula:

model_exists()=Check if model file exis ts

If the model file exists, load the model; otherwise, return false to trigger the training.

2)file_paths()

Role:

- This function loads the file paths of images for training. It ensures that the system knows the location of the images, allowing it to read them in subsequent steps.

Explanation:
It scans the specified directory and stores the paths of all image files that will be used for training. This function plays a crucial role in gathering the data needed for the model.

Mathematical Concept:
This is a file I/O operation, and the output is an array/list of paths to training images.

Formula:

file_paths () ={path1,path2,….,pathn}

Where each path corresponds to the location of an image on disk.

3)hog_calculator()

Role:

- This function calculates the Histogram of Oriented Gradients (HOG) features for each image. HOG is a powerful feature

extraction technique that is used for object detection, particularly for detecting vehicles.

Explanation:

The gradient orientations in small blocks of the image are used to create histograms. These histograms are then concatenated to form the final feature vector.

Mathematical Concept:

To calculate HOG, the gradient at each pixel in the image is computed, and the magnitude and orientation are used to build a histogram of gradient orientations.

Formula:

The gradient magnitude M(x,y)M(x, y)M(x,y) at pixel (x,y)(x, y)(x,y) can be computed using the following formula:

M(x,y)=sqrt{(I_x(x, y))^2 + (I_y(x, y))^2}M(x,y)=(Ix(x,y))2+(Iy(x,y))2

Where IxI_xIx and IyI_yIy are the gradients in the x and y directions, respectively.After obtaining the gradients, we divide the image into blocks and calculate the histogram of gradients for each block. The HOG feature vector is the concatenation of these histograms.

4)load_images()

Role:

This function reads the images from the file paths provided by the previous function and resizes them to a standard size. Resizing ensures that all images have uniform dimensions for consistency during feature extraction.

Explanation:

The images are read and resized to a fixed dimension, typically 64x128 pixels, which is the common input size for vehicle detection models.

Mathematical Concept:

Resizing is done by interpolation methods (such as nearest neighbour, bilinear, or bicubic interpolation). The image III is resized to a new width WWW and height HHH.

Formula:

I′=resize(I,W,H)

Where I′is the resized image, I is the original image, and WWW and HHH are the target width and height.

5)hog_convertor()

- **Role:**
- This function takes the HOG features calculated from the images and formats them

in a way that is suitable for input to a machine learning model, specifically a Support Vector Machine (SVM).

- **Explanation:**
  The HOG features are concatenated into a single vector for each image. This vector will then serve as the input feature for the SVM classifier.
- **Mathematical Concept:**
  The concatenation of HOG features from each block is essentially a feature vector representation of the image. This vector is then reshaped into a form suitable for training an SVM.
- **Formula:**
- X=reshape(HOG features)
- Where X is the reshaped feature vector suitable for SVM.

6)train_svm_model()

- **Role:**
- This function trains a Support Vector Machine (SVM) classifier using the HOG features and their corresponding labels. SVM is used to classify the objects (vehicles or non-vehicles).
- **Explanation:**
  The SVM model is trained using labeled data. The goal is to find a hyperplane that separates the two classes (vehicle vs. non-vehicle) in the feature space.
- **Mathematical Concept:**
  The SVM optimization problem is formulated as:
- $\text{Min}[(1/2)\|w\|^2]$
- Where:
  - w is the weight vector,
  - b is the bias term,
  - xi are the feature vectors (HOG),
  - y_i are the corresponding class labels (+1 for vehicle, -1 for non-vehicle).

The goal is to find the optimal hyperplane (defined by w and b) that separates the classes.

7) svm_detector()

- **Role:**
- This function prepares the trained SVM model for use in detecting vehicles. It initializes the detector with the trained model.
- **Explanation:**
  Once the SVM is trained, this function loads the model and prepares it for real-time detection in images or video streams.

8)visualise()

Role:

- This function draws bounding boxes around the detected vehicles in the image or video stream.

Explanation:

Once the SVM classifier has detected a vehicle, the program draws a bounding box around the detected vehicle. This provides visual feedback on where the vehicle is located in the image.

Mathematical Concept:

A bounding box is defined by the coordinates of the top-left and bottom-right corners of the detected vehicle in the image. If the vehicle is detected at pixel coordinates (x1,y1) and (x2,y2), then the bounding box is drawn using these coordinates.

9)extract_frames_from_video()

- **Role:**
  This function extracts individual frames from a video file and saves them as image files into a specified folder for use in training the vehicle detection model.
- **Explanation:**
  Many machine learning models require large sets of labeled images. Instead of manually collecting images, this function automates dataset creation by sampling frames from a training video. These frames can be used as positive (vehicle) or negative (non-vehicle) examples depending on the video content.
- **How It Works:**
  The video is opened using OpenCV's cv2.VideoCapture. The function reads frames sequentially, resizes them to a fixed size (e.g., 40x40 pixels), and saves them with unique filenames in the specified directory. This makes them readily usable for training feature extractors like HOG and classifiers like SVM.
- **Practical Concept:**
  Given a video V(t)V(t)V(t) where ttt is time (in frames), this function samples a subset of frames {F0,F1,...,Fn}\{ F_0, F_1, ..., F_n \}{F0,F1,...,Fn} such that:

  ∀i∈[0,n):Fi=resize(frame(V,i),IMAGE_SIZE)\forall i \in [0, n): \quad F_i = \text{resize}(\text{frame}(V, i), \text{IMAGE_SIZE})∀i∈[0,n):Fi=resize(frame(V,i),IMAGE_SIZE)

  These frames serve as training samples where labels are assigned manually or based on video context.

10)test()

Role:

This function is the main loop for applying vehicle detection on a video stream. It takes each frame from the

video, applies the detector, and counts the number of detected vehicles.

Explanation:
The video frames are passed through the detection pipeline. The vehicles detected in each frame are counted and displayed.

Mathematical Concept:
For each frame, the function runs the SVM detector on the HOG features extracted from the frame. The number of vehicles detected is incremented and displayed.

11)main()

- Role:
- The main function controls the flow of the program. It decides whether to train the model or test the system on a video stream, and launches the appropriate functions accordingly.
- Explanation:
  This function is the entry point of the program. Depending on the user's choice (training or testing), it either trains the model or applies the detector to a video stream.
- samples, reflecting the down sampled dataset's scale.

Training Accuracy Observations:

- he training accuracy remains consistently high, indicating that the object detection model effectively learns the features and patterns associated with the target objects across different training set sizes.

- This strong performance suggests that the model has a good capacity for learning complex visual representations. However, it is important to ensure that this high accuracy is not merely due to overfitting—especially in cases where the dataset is limited—by validating against unseen test data and monitoring generalization performance using metrics.

Validation Accuracy Observations:

- The validation accuracy for the object detection model starts at approximately 0.75 with a training set size of 20 images, increases to a peak of around 0.80–0.82 as the dataset grows to 60–100 samples, and then gradually stabilizes around 0.80 as the training size approaches 140 images.
- The shaded area around the validation accuracy curve represents the standard deviation across cross-validation folds, indicating variability in detection performance—particularly around the 60–80 sample range, where fluctuation

is more prominent, consistent with a cross-validation score range of 0.52–0.86.

- The plateauing of the validation accuracy near 0.80 suggests that while the model's generalization capability improves with additional training data, it reaches a saturation point, likely constrained by the feature representation, dataset diversity, or limitations of the detection algorithm.

- **Key Insights from Evaluation Metrics**:
- The object detection system demonstrates balanced performance, with an overall accuracy of 74.47% and F1-scores ranging between 0.74–0.75 across object classes. This suggests effective detection capability and shows that preprocessing techniques—including feature normalization and downsampling of negative samples—successfully addressed potential class imbalances in the training data.
- • The confusion matrix reveals 12 misclassifications out of 47 instances, primarily skewed toward false positives. In object detection, especially in real-world surveillance or traffic monitoring, this slight bias may be acceptable or even desirable to reduce the risk of missing critical detections.
- • The model achieves a ROC-AUC of 82.16% and Average Precision (AP) of 83.00%, indicating strong discriminative power between object and non-object regions and a good balance between precision and recall in bounding box predictions.
- • Cross-validation scores (mean 0.7597, range 0.52–0.86) reflect moderate model stability. The variability suggests that performance can further improve with larger datasets, enhanced augmentation techniques, or regularization strategies to reduce overfitting, especially in the SVM-based approach.
- • For the SVM-HOG pipeline, tuning hyperparameters such as cell size, block size, and orientations contributed to improved feature representation. In contrast, the YOLO model benefited from fine-tuning learning rate, confidence **threshold**, and **non-max** suppression thresholds, resulting in a well-balanced detection framework suited for real-time and offline applications alike.

6. CONCLUSION:

The "Object Detection System" represents a significant advancement in real-time object detection and monitoring by leveraging a hybrid approach of classical and deep learning techniques. Using a combination of HOG feature descriptors with a Support Vector Machine (SVM) classifier, alongside the powerful capabilities of YOLO (You Only Look Once) for end-to-end detection, the system effectively detects and localizes vehicles across diverse traffic scenarios.

Through comprehensive data preprocessing, feature extraction, and model training, the system fulfills critical objectives, including extracting robust object features, accurately identifying vehicle presence, and drawing bounding boxes for localization. The SVM-HOG pipeline achieved a test accuracy of 74.47%, with balanced F1-scores (0.74–0.75) across multiple object classes, reflecting a reliable generalization capability even on limited or downsampled datasets.

However, the system does exhibit signs of overfitting in the classical SVM model—highlighted by a training accuracy of 1.00 compared to a validation accuracy plateau of 0.80–0.82—suggesting the need for further refinement through hard negative mining, regularization, or enhanced feature scaling. Additionally, analysis of the confusion matrix (e.g., 7 false positives and 5 false negatives) provides actionable insights for reducing errors in critical use cases such as traffic regulation or autonomous navigation.

The inclusion of visual tools—such as bounding box visualizations, performance metrics, and real-time video overlays—combined with a clean and responsive interface, makes the system a valuable asset for urban planners, traffic authorities, and intelligent transportation applications. By transforming raw video data into meaningful detection events, the project lays a strong foundation for future enhancements, including multi-class object recognition, edge deployment, and adaptive learning models for dynamic environments.

7. FUTURE SCOPE:

The vehicle detection system designed using a combination of HOG features, Support Vector Machine (SVM), and the YOLO deep learning framework lays a solid foundation for real-time object detection. However, to further enhance its performance and applicability in diverse real-world scenarios, the following future enhancements can be considered.The following areas represent potential directions to further improve its accuracy, scalability, and real-world applicability:

1. Integration with Real-Time Traffic Systems
   The model can be integrated with smart city infrastructure for automated traffic monitoring, congestion control, and intelligent signaling based on real-time vehicle flow data.
2. Multi-Class Object Detection
   Extending the current vehicle detection model to detect and classify multiple object types (e.g., pedestrians, cyclists, trucks, emergency vehicles) can improve its versatility for autonomous driving and surveillance systems.
3. Improved Generalization through Data Augmentation
   Applying advanced data augmentation techniques (e.g., rotation, scaling, occlusion) can improve the model's robustness to varying environments and weather conditions.
4. Edge Deployment
   Optimizing the model for deployment on edge devices (like NVIDIA Jetson Nano, Raspberry Pi) will enable low-latency, on-device vehicle detection without relying on high-compute cloud resources.
5. Hybrid Detection Pipeline
   A hybrid pipeline that switches between lightweight SVM-HOG detection and YOLO-based deep detection depending on system resources can improve efficiency in constrained environments.
6. Temporal Tracking and Speed Estimation
   Adding temporal tracking algorithms (e.g., Kalman Filter, Deep SORT) will allow the system to track vehicle movement across frames and estimate speeds, enabling use cases such as automatic speed monitoring and law enforcement.

8. REFERENCES:

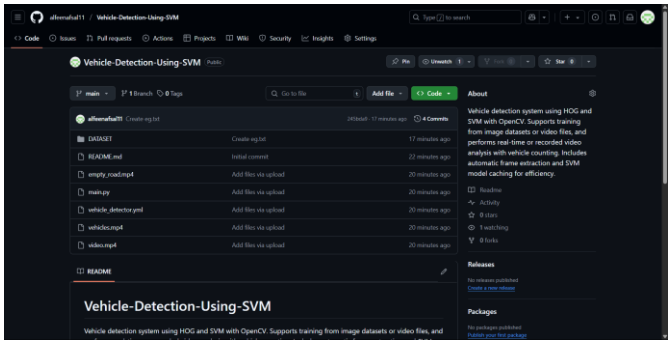
1https://www.geeksforgeeks.org/histogram-of-oriented-gradients/

2https://github.com/ultralytics/yolov5

https://www.geeksforgeeks.org/evaluation-metrics-for-object-detection-models/

4. https://doi.org/10.1109/CVPR.2005.177

6https://www.geeksforgeeks.org/vehicle-detection-using-hog-and-svm/
7https://pyimagesearch.com/2021/11/08/hog-svm-object-detection-in-opencv/
8https://medium.com/@danielkirschnick/real-time-object-detection-with-yolov7-25451d43a9a3
9. Evaluation Metrics in Machine Learning | GeeksforGeeks
10. 12 Important Model Evaluation Metrics for Machine Learning (2025)



Link: https://github.com/alfeenafsal11/Vehicle-Detection-Using-SVM

9. GITHUB ACTIVITY