OBJECT DETECTION AND TRAFFIC SIGN RECOGNITION USING UP-DETR FOR AUTONOMOUS VEHICLES

A PROJECT REPORT

Submitted by

SRIVATHSAVA M

(2023176034)

A report for the dissertation-I
submitted to the faculty of

INFORMATION AND COMMUNICATION ENGINEERING

in partial fulfillment
for the award of the degree
of

MASTER OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

SPECIALIZATION IN AI & DS

DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING GUINDY
ANNA UNIVERSITY
CHENNAI 600 025

JAN 2025

ANNA UNIVERSITY

CHENNAI - 600 025

BONAFIDE CERTIFICATE

Certified that this project report titled **OBJECT DETECTION AND TRAFFIC SIGN RECOGNITION USING UP-DETR FOR AUTONOMOUS VEHICLES** is the bonafide work of **SRIVATHSAVA M (2023176034)** who carried out project work under my supervision. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or an award was conferred on an earlier occasion on this or any other candidate.

PLACE:CHENNAI
DATE:

Dr. M. VIJAYALAKSHMI
PROFESSOR
PROJECT GUIDE
DEPARTMENT OF IST, CEG
ANNA UNIVERSITY
CHENNAI 600025

COUNTERSIGNED

Dr. S. SWAMYNATHAN
HEAD OF THE DEPARTMENT
DEPARTMENT OF INFORMATION SCIENCE AND TECHNOLOGY
COLLEGE OF ENGINEERING GUINDY
ANNA UNIVERSITY
CHENNAI 600025

ABSTRACT

Traditional object detection models like Faster R-CNN and YOLO have been pivotal in autonomous vehicle perception but face challenges in dynamic environments. These include difficulties with small object detection, occlusions, and reliance on computationally expensive pipelines. Moreover, their dependency on large annotated datasets and inability to generalize across diverse conditions, such as varying lighting and weather, limits their effectiveness for real-time autonomous driving applications.

This project proposes a object detection system based on the Unsupervised Pre-training for Detection Transformer (UP-DETR). By employing a transformer-based architecture and unsupervised pre-training with random query patch detection, the model reduces reliance on labeled data while enhancing generalization. Pre-training is conducted on the Tiny ImageNet dataset with a frozen ResNet backbone, followed by fine-tuning on the COCO dataset for accurate object detection and traffic sign recognition. Validation using video frames from the CARLA simulator ensures robustness under real-world driving scenarios.

The proposed system achieves faster convergence, reduced data dependency, and reliable detection performance even in challenging conditions like poor lighting and occlusions.Testing in CARLA demonstrates the system's adaptability and real-time capability, achieving IoU values of 0.50 to 0.95, Precision between 0.216 and 0.624, and F1 Scores ranging from 0.460 to 0.824. These metrics highlight its reliability in detecting and recognizing objects accurately while maintaining efficient performance.

ABSTRACT TAMIL

ACKNOWLEDGEMENT

It is my privilege to express my deepest sense of gratitude and sincere thanks to **Dr. M. VIJAYALAKSHMI,** Professor, Project Guide, Department of Information Science and Technology, College of Engineering, Guindy, Anna University, for her constant supervision, encouragement, and support in my project work. I greatly appreciate the constructive advice and motivation that was given to help me advance my project in the right direction.

I am grateful to **Dr. S. SWAMYNATHAN,** Professor and Head, Department of Information Science and Technology, College of Engineering Guindy, Anna University for providing us with the opportunity and necessary resources to do this project.

I would also wish to express my deepest sense of gratitude to the Members of the Project Review Committee: **Dr. S. SRIDHAR,** Professor, **Dr. G. GEETHA,** Associate Professor, **Dr. D. NARASHIMAN,** Teaching Fellow Department of Information Science and Technology, College of Engineering Guindy, Anna University, for their guidance and useful suggestions that were beneficial in helping me improve my project.

I also thank the faculty member and non teaching staff members of the Department of Information Science and Technology, Anna University, Chennai for their valuable support throughout the course of our project work.

SRIVATHSAVA M

(2023176034)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

UP-DETR	Unsupervised Pre-Training for Detection Transformers
YOLO	You Only Look Once
COCO	Common Objects in Context Dataset
SSD	Single Shot Detection
AV	Autonomous Vehicles
ViT	Vision Transformers
CNN	Convolutional Neural Networks
AD	Autonomous Driving
RPN	Region Proportional Networks
IOU	Intersection Over Union
GPT	Generative Pretrained Transformer
SOTA	State-of-the-art

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

In recent years, object detection and traffic sign recognition [1] have become critical areas of study in computer vision due to their relevance in autonomous driving systems [2]. Accurate object detection [3] enables vehicles to identify and track other vehicles, pedestrians, and obstacles, while traffic sign recognition helps autonomous systems understand and comply with traffic regulations, such as speed limits, stop signals, and yield signs.

Traditional object detection models, like Faster R-CNN [4], involve multi-stage processes that rely on region proposal networks (RPNs) and post-processing techniques such as Non-Maximum Suppression (NMS). These approaches, while effective, are complex and computationally intensive, which can hinder the real-time performance required by autonomous vehicles. DETR (Detection Transformer) [5] addresses these issues by reframing object detection as a set prediction problem, using transformer-based self-attention mechanisms to learn spatial and contextual relationships between objects. This end-to-end architecture eliminates the need for region proposals and NMS, streamlining the detection process.

However, DETR requires large-scale annotated datasets, such as COCO, and extensive training to achieve optimal performance. These requirements present practical challenges, as autonomous driving demands robust performance in diverse, real-world traffic scenarios where data labeling is both resource-intensive and time-consuming. This study explores UP-DETR

[6] as a solution, utilizing unsupervised pre-training with random query patch detection to reduce the reliance on labeled data and improve the model's ability to detect objects and recognize traffic signs in autonomous driving environments.

1.1.1 Autonomous Vehicles and Object Detection in Real-World Applications

Object detection is a cornerstone technology for autonomous vehicles [2], where the ability to identify and locate objects in real-time is crucial for safe navigation. AVs rely on a combination of sensors, such as cameras, LiDAR, and radar, to perceive their environment and make decisions. However, the key challenge lies in processing this sensor data efficiently to detect various objects, including pedestrians, other vehicles, traffic signs, and obstacles, especially in dynamic environments.

In traditional object detection models like Faster R-CNN [4] and YOLO [7], the focus is on detecting objects in static environments, usually with high-quality, well-lit images. However, real-world driving scenarios often present more complexities, such as poor lighting, adverse weather conditions, occlusions, and dynamic objects. These factors make detecting objects in real-time more challenging.

1.1.2 Object Detection for AVs: The Importance of Efficiency and Accuracy

For AVs, object detection must be both fast and accurate to make timely decisions [8]. YOLO models are often favored in real-time applications due to their speed, processing images in a single pass. However, the accuracy

of YOLO can drop in crowded or complex environments where small or overlapping objects are difficult to detect.

On the other hand, transformer-based models like DETR [5] provide a more global understanding of the image, which is essential for accurately detecting objects in cluttered or partially occluded environments. By using self-attention mechanisms [8], transformers can model the relationships between objects and their context within the scene, making them particularly suitable for the complexity of real-world driving environments.

1.1.3 UP-DETR

The UP-DETR [6] model is composed of a carefully designed architecture to effectively perform object detection tasks by leveraging transformers and a convolutional backbone. At its core, the backbone network is a ResNet-based CNN responsible for extracting high-level features from the input frames. This backbone processes raw image data and produces hierarchical feature maps that serve as input to the transformer.

Following this, the Transformer [9] Encoder-Decoder architecture plays a central role in the detection pipeline. The encoder captures and encodes the spatial relationships present in the feature maps, enabling the model to understand the global context of objects and their surrounding environment. The decoder processes these encoded features and interacts with object queries—a set of learnable embeddings that represent potential object locations in the image. These queries allow the model to directly predict both bounding boxes for object localization and the corresponding class labels for object recognition.

By combining the feature extraction power of ResNet [10] with the global reasoning capabilities of the transformer, UP-DETR achieves a

streamlined end-to-end object detection process that eliminates traditional hand-crafted components like region proposals and anchor boxes. This architecture enables the model to learn spatial relationships effectively and efficiently predict objects in complex scenes.

1.1.4 Role of UP-DETR in Autonomous Vehicles

In our work, UP-DETR plays a pivotal role in improving the efficiency and generalization of object detection systems for AVs. By using unsupervised pre-training, UP-DETR [6] can adapt to dynamic and varying conditions in real-world scenarios, learning from unlabeled data and reducing the reliance on extensive labeled datasets. This is particularly useful in autonomous driving, where data annotation is costly and time-consuming.

Moreover, UP-DETR's transformer-based architecture [9] enables better handling of complex scenes that involve overlapping objects, occlusions, and dynamic elements, all of which are common in AV applications. As a result, the model can provide more accurate and context-aware object detection, which is crucial for AVs to make safe and reliable driving decisions.

The UP-DETR model is specifically designed to enhance object detection for autonomous vehicles by incorporating both transformer-based reasoning and unsupervised pre-training. This ensures that the model can handle diverse driving environments with complex and dynamic objects, leading to improved accuracy in detection tasks such as recognizing pedestrians, vehicles, traffic signs, and other road obstacles.

By fine-tuning UP-DETR on domain-specific datasets like COCO , you can create a model that is not only accurate but also highly efficient in real-world applications, providing autonomous vehicles with the necessary

perception capabilities to navigate safely and effectively.

1.2 PROBLEM STATEMENT

- Autonomous vehicles operate in dynamic road conditions with rapidly changing obstacles, lighting, and weather. Current detection models often struggle to adapt to these variables, resulting in failures to identify critical objects and traffic signs accurately.
- Real-time detection and recognition of traffic signs, such as stop signs and speed limits, are vital for safe navigation. However, inconsistencies in recognition under varying conditions pose safety risks, as missed or misinterpreted signs can lead to hazardous outcomes.
- Fluctuating road scenarios, such as construction zones, road debris, or sudden changes in traffic flow, require reliable and real-time detection systems to enable effective responses.

1.3 OBJECTIVES OF THE PROJECT

- To implement an Unsupervised Pre-Training framework for DETR that leverages unlabeled data to improve feature representation and enhance the model's detection performance.
- To develop a robust and efficient object detection and traffic sign recognition system using UP-DETR, focusing on accurately identifying and classifying objects and signs relevant to autonomous driving scenarios.

- To enhance both detection accuracy and processing speed, ensuring the system's reliability and responsiveness for real-time applications in autonomous vehicles, enabling safe and efficient navigation in complex environments.

1.4 SCOPE OF THE PROJECT

- The study focuses on the unsupervised pre-training of the transformer decoder within the DETR framework. Random query patch detection will be implemented to train the decoder on unlabeled data, preparing it for object detection and traffic sign recognition tasks.
- The performance of UP-DETR will be evaluated on standard benchmarks, such as COCO , using metrics like IOU, training time, and convergence rate. These evaluations will particularly focus on the model's applicability to autonomous driving by testing its ability to detect and recognize objects and traffic signs in diverse conditions.

1.5 ORGANIZATION OF THE REPORT

This report is organized into 6 chapters, describing each part of the project with detailed illustrations and system design diagrams.

CHAPTER 2: Literature Review reviews existing research, studies, and relevant literature related Detection Models. Discusses the background, theories, and methodologies used by other researchers

CHAPTER 3: System Design describes the design of the project. Explains the architecture, components, algorithms, and any other technical details.

CHAPTER 4: Implementation provides details about how the project was implemented. Discusses the outcome of the modules.

CHAPTER 5: Result and Analysis presents the results of the project. Analyzes the outcomes, compare them with expectations, and discuss any challenges faced during implementation.

CHAPTER 6: Conclusion and Future Work summarizes the findings and draws conclusions. Discusses the significance of your work and its implications

CHAPTER 2

LITERATURE SURVEY

2.1 TRANSFORMERS

Transformers, introduced by Vaswani et al. [9] in ”Attention Is All You Need,” revolutionized machine learning by replacing sequential processing in RNNs and LSTMs with a self-attention mechanism, allowing efficient processing of entire sequences. Originally designed for NLP tasks like machine translation, transformers quickly became the foundation for models such as BERT and GPT, which showcased their scalability and adaptability.

The application of transformers to computer vision began with ViT Dosovitskiy et al.[11], which treated images as sequences of non-overlapping patches, using self-attention to capture global context. However, ViT's dependency on large datasets highlighted the need for more efficient methods. This was addressed by DeiT (Data-efficient Image Transformer) Touvron et al.[12], which incorporated distillation tokens to reduce the need for massive pre-training.

Further advancements like the Swin Transformer Liu et al.[13] introduced a hierarchical structure to combine the benefits of CNNs and transformers, improving performance in object detection and segmentation tasks. DETR by Carion et al.[5] utilized a transformer-based encoder-decoder framework to unify object detection and segmentation into an end-to-end system, demonstrating the versatility of transformers for spatial reasoning tasks.

The self-attention mechanism and ability to model global dependencies make transformers ideal for object detection in autonomous

vehicles, where spatial relationships are crucial. Models like DETR and UP-DETR align with this work, offering efficient solutions for real-world tasks, and innovations like the Swin Transformer and DeiT provide insights for optimizing transformer-based architectures.

2.2 DETECTION TRANSFORMER

The landscape of object detection has been transformed by advancements in deep learning, beginning with traditional multi-stage models like Faster R-CNN Ren et al.[4]. Faster R-CNN introduced the Region Proposal Network (RPN), which efficiently proposed regions of interest (RoIs) for classification and refinement. However, these models were complex, relying on multiple stages and post-processing steps such as Non-Maximum Suppression (NMS) to handle overlapping bounding boxes.

In 2020, Carion et al.[5] introduced DETR (DEtection TRansformer), a paradigm-shifting framework that simplified object detection with a single-stage transformer-based architecture. Unlike traditional methods that depend on region proposals, anchor boxes, or NMS, DETR treats object detection as a set prediction problem. It uses a transformer encoder-decoder to output a fixed set of object predictions, where each object is represented by a learned query. The transformer decoder predicts bounding box positions and class labels in parallel while capturing global relationships between objects and their context through the self-attention mechanism.

DETR's key innovations includes End-to-End Learning, Global Context Modeling, Unified Framework

This end-to-end approach eliminates the need for post-processing and simplifies the detection pipeline, marking a significant evolution in object

detection models.

2.3 UNSUPERVISED PRE-TRAINING

Object detection often struggles with the reliance on large labeled datasets, which are expensive and time-consuming to annotate. To address this, unsupervised pre-training has emerged as a promising approach, where models first learn features from unlabeled data and are later fine-tuned on smaller labeled datasets. Early methods, such as autoencoders and generative models, faced challenges in learning meaningful features for object detection. However, breakthroughs in contrastive learning, particularly SimCLR Chen et al.[14], enabled self-supervised learning by maximizing similarity between augmented views of the same image.

Further advancements like MoCo He et al.[15] improved efficiency by introducing memory banks to store negative samples, enabling training with smaller batch sizes. These techniques laid the foundation for unsupervised pre-training in object detection tasks.

Building on this, UP-DETR Dai et al.[6] introduced random query patch detection as a pre-training task. Instead of labeled data, the transformer decoder learns to predict the location of randomly cropped patches within an image. This approach allows UP-DETR to overcome the slow convergence of traditional models like DETR and enhances generalization, especially when fine-tuned on datasets like COCO.

UP-DETR is particularly advantageous for autonomous vehicle applications, where labeled data is limited, and real-time performance is critical. By learning spatial relationships and object localization without labels, UP-DETR addresses data dependency challenges. The random query patch

detection task improves generalization, reduces training time, and ensures the model performs efficiently in dynamic real-world environments.

Drawing on techniques from SimCLR and MoCo, UP-DETR represents a step toward data-efficient object detection, offering strong performance on smaller datasets. Its ability to generalize across tasks makes it a powerful solution for detecting and localizing diverse objects in autonomous driving, where accurate real-time detection is essential for safety and navigation.

2.4 FINE-TUNING

Fine-tuning is a vital technique in deep learning, enabling the transfer of knowledge from pre-trained models to specific tasks. Instead of training from scratch, models initialized with weights learned on large datasets like ImageNet are fine-tuned on smaller, task-specific datasets, reducing the need for extensive labeled data and computational costs.

The success of AlexNet Krizhevsky et al.[16] demonstrated the effectiveness of CNNs, while He et al.[17] showed that fine-tuned models, such as ResNet-50, outperform models trained from scratch, even on smaller datasets. Fine-tuning typically involves freezing initial layers (to preserve general features like edges and textures) while training deeper layers to adapt to task-specific features, reducing overfitting.

In object detection, models like Faster R-CNN Ren et al.[4] and YOLO Redmon et al.[7] have benefited from fine-tuning on datasets like COCO. For specialized applications, such as autonomous vehicles, fine-tuning UP-DETR on domain-specific datasets like COCO enhances the model's ability to detect pedestrians, vehicles, and traffic signs in dynamic environments.

In UP-DETR, the ResNet-50 backbone—pre-trained on ImageNet—extracts general features, while fine-tuning focuses on the transformer decoder to learn task-specific object localization and recognition. This process ensures the model can generalize to real-world scenarios, improving performance in diverse and dynamic environments critical for autonomous driving systems.

2.5 BACKBONE RESNET-50

The backbone network is a critical component in any deep learning model, as it is responsible for extracting meaningful features from input images. ResNet-50, introduced by He et al.[10], is one of the most popular backbones used in modern computer vision models. ResNet-50 employs residual connections to combat the vanishing gradient problem, allowing for deeper networks to be trained effectively.

ResNet-50 is structured with bottleneck blocks that use 1x1 convolutions for dimensionality reduction, followed by 3x3 convolutions for feature extraction, and another 1x1 convolution to restore dimensionality. These layers enable the model to capture both fine-grained and abstract features from images.

In object detection models like UP-DETR, ResNet-50 acts as a powerful feature extractor. It processes raw image data, providing the transformer decoder with rich, high-level features that are crucial for accurate object localization and classification.

In UP-DETR, ResNet-50 provides the essential feature extraction required for the transformer decoder to perform object detection efficiently. By leveraging pre-trained ResNet-50 models, your system can achieve high

accuracy with minimal fine-tuning, making it highly suitable for autonomous vehicle applications where real-time performance is crucial.

2.6 RANDOM QUERY PATCHING

Random query patching is a unique unsupervised pre-training task introduced in UP-DETR Dai et al.[6], designed to reduce the dependency on large annotated datasets. The task involves randomly cropping patches from an image and training the model's transformer decoder to predict the locations of these patches within the original image. This pre-training task helps the model learn spatial relationships, object localization, and context, all without requiring labeled data.

The core idea behind random query patching is that the model can learn useful features from unlabeled images by simply understanding the spatial arrangement of patches. This enables the transformer decoder to develop a deeper understanding of the structure of objects and their relationships in a scene, which is vital for tasks like object detection and scene segmentation.

This approach helps UP-DETR overcome some of the challenges faced by traditional object detection models, such as the need for extensive labeled datasets and long training times. By training on randomly cropped patches, the model can efficiently pre-train on unlabeled data, making it more adaptable to tasks with limited annotated examples.

Random query patching in UP-DETR directly supports your project's goal of reducing the need for large labeled datasets in object detection tasks. This method allows the model to learn critical features and spatial relationships, enabling it to perform well when fine-tuned on smaller, domain-specific datasets like KITTI for autonomous driving.

2.7 OBJECT DETECTION

Object detection is a fundamental challenge in computer vision, combining localization (predicting bounding boxes) and classification (assigning labels). Unlike image classification, it requires identifying multiple objects within an image.

Early methods relied on handcrafted features, such as HOG (Histogram of Oriented Gradients) combined with SVMs , which were effective for simple tasks like pedestrian detection but struggled with occluded objects and were computationally intensive.

The introduction of Region-based CNNs (R-CNNs) Girshick et al.[18] combined Selective Search for region proposals with CNNs for detection, significantly improving accuracy. However, R-CNN remained computationally expensive as it required running CNNs on multiple region proposals.

2.7.1 End-to-End Architecture

To streamline the pipeline, YOLO Redmon et al.[7] reframed object detection as a regression task, dividing the image into grids and predicting bounding boxes and class labels in a single pass. YOLO achieved real-time detection speeds but struggled with small object accuracy.

SSD (Single Shot Multibox Detector) Liu et al.[19] further improved efficiency by introducing multi-scale feature maps, enabling better detection of objects of varying sizes. Despite these advancements, anchor-based detectors like YOLO and SSD faced challenges with detecting objects that had complex shapes or unusual aspect ratios.

These innovations laid the groundwork for modern end-to-end object detection models, balancing speed and accuracy for real-world applications.

2.8 NON MAXIMUM SUPPRESSION

Non-Maximum Suppression (NMS) has long been a critical post-processing step in traditional object detection pipelines. NMS is used to eliminate redundant bounding box predictions that often occur when the model detects the same object multiple times, each with different bounding boxes. These redundant predictions are typically the result of the model assigning high confidence scores to multiple boxes that overlap for the same object.

NMS works by sorting the bounding boxes based on their confidence scores and selecting the box with the highest score. The other boxes that overlap significantly (typically measured using the Intersection over Union (IoU) metric) are discarded. This process helps ensure that only the best prediction for each object remains.

While NMS is effective, it has its limitations like Greedy Nature, Inconsistent handling of Overlapping objects, Inefficiency with small objects.

In response to these issues, variations of NMS have been proposed to improve its performance in challenging scenarios. Soft-NMS, introduced by Bodla et al.[20], is one such improvement. Unlike traditional NMS, which completely removes overlapping boxes based on a threshold, Soft-NMS decays the scores of overlapping boxes instead of discarding them outright. This allows for better handling of objects in close proximity and reduces the risk of removing true positive detections.

2.9 SUMMARY OF EXISTING SYSTEM

The evolution of object detection has progressed from traditional handcrafted methods, such as HOG and SVM, to deep learning models that automate feature extraction and improve accuracy. AlexNet Krizhevsky et al.[16] marked a breakthrough, followed by advancements like VGGNet and ResNet, with ResNet-50 widely adopted for its ability to capture rich features. Faster R-CNN Ren et al.[4] introduced Region Proposal Networks (RPN) for better detection speed but relied on Non-Maximum Suppression (NMS) to handle redundant bounding boxes. YOLO Redmon et al.[7] simplified detection with a single-pass approach, enhancing speed but struggling with small objects. DETR Carion et al.[5] revolutionized detection by using a transformer-based architecture for set prediction, capturing global relationships, but faced challenges with slow convergence and small object detection. UP-DETR Dai et al.[6] addressed these issues with unsupervised pre-training using random query patch detection, enhancing generalization and reducing reliance on labeled data. Inspired by unsupervised methods like SimCLR and MoCo, UP-DETR leverages fine-tuning to adapt models like ResNet-50 for specific tasks like traffic sign recognition. Innovations like Soft-NMS further refine detection for small or occluded objects. For autonomous vehicles, detection systems like DETR and UP-DETR effectively handle global dependencies, enabling fast, accurate, and data-efficient solutions for safe navigation in dynamic environments.

CHAPTER 3

SYSTEM DESIGN

This chapter deals with system design of Object Detection and Traffic Sign Recognition using UP-DETR for Autonomous Vehicles with overall architecture and its detailed description.

The system design, shown in Fig 3.1, uses the UP-DETR model for Object Detection and Traffic Sign Recognition in Autonomous Vehicle applications. The process starts with video data from the CARLA Simulator, which simulates real-world driving environments, providing essential data for detection tasks.

After video acquisition, frame extraction is performed to break the video into individual frames for processing. These frames are analyzed for objects such as vehicles, pedestrians, and traffic signs.

The pre-training phase involves training the UP-DETR model on a large, unsupervised dataset to learn general object features. The model is then fine-tuned on labeled dataset to specialize in detecting traffic signs and objects relevant to autonomous driving.

Once fine-tuning is complete, the model is deployed for real-time detection, processing frames to detect objects and signs for vehicle navigation. This ensures the system operates efficiently and accurately in dynamic driving scenarios with minimal latency.

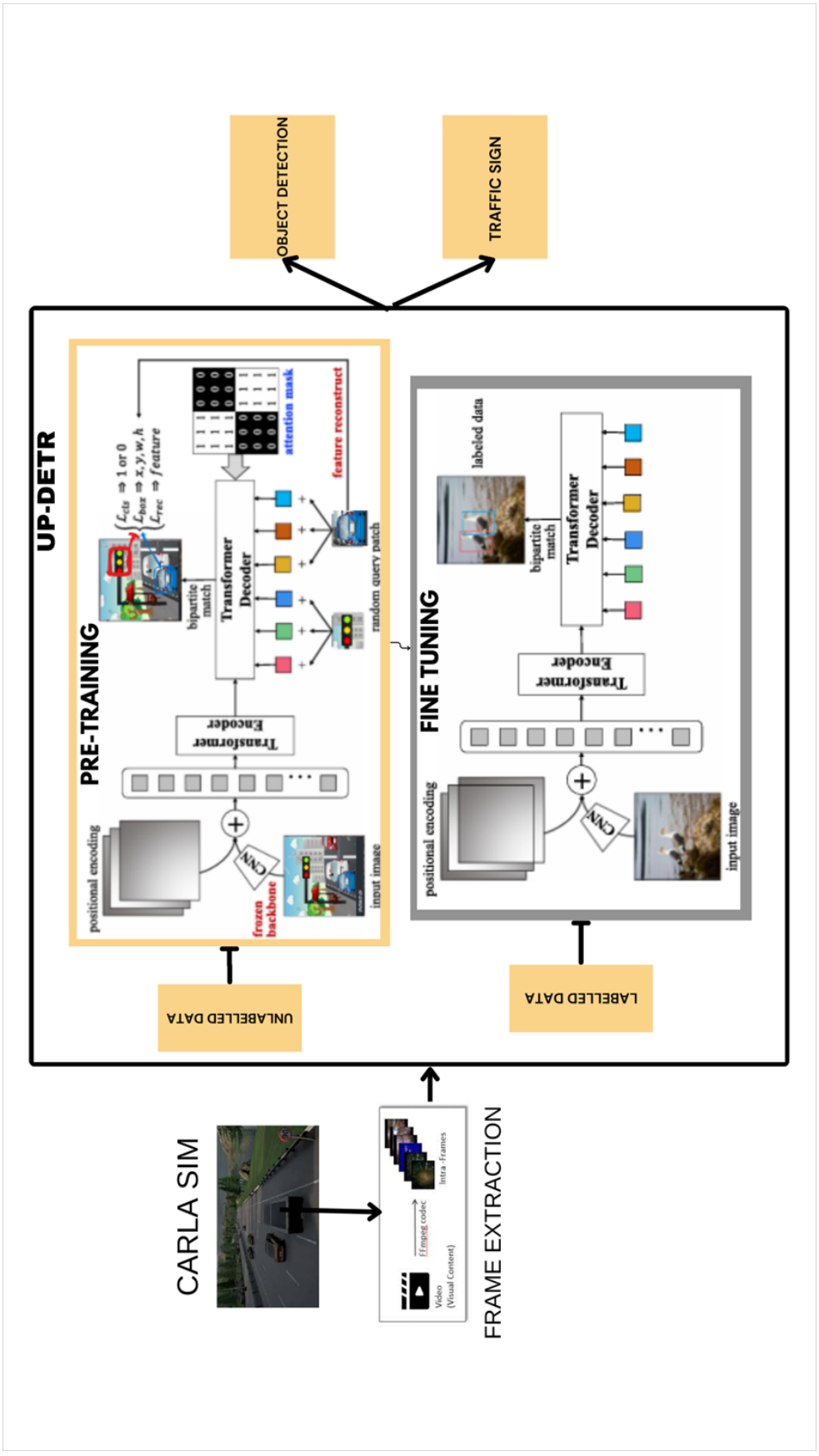

Figure 3.1: System Architecture for Object Detection and Traffic Sign Recognition using UP-DETR

3.1 INPUT DATA

This project uses video frames from the CARLA simulator, capturing diverse traffic scenarios with vehicles, pedestrians, and traffic signs. These frames are preprocessed to meet UP-DETR input requirements, providing realistic and challenging data for training and evaluation.

3.1.1 Video Data from CARLA Simulator

Figure 3.2: Carla Simulator

The CARLA simulator provides synthetic driving videos that simulate various traffic conditions and environments which you can see in Fig 3.2. These videos contain realistic road scenarios with other vehicles, pedestrians, and traffic signs, making them ideal for testing and training autonomous vehicle models.

3.1.2 Frame Extraction

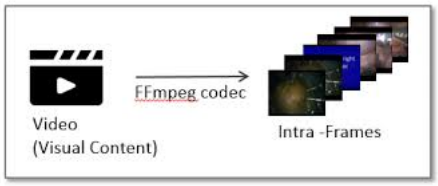

Figure 3.3: Frame Extraction

Frames are extracted from videos at specified intervals based on the desired frame rate (e.g., 1 frame per second or higher for faster processing) like

in Fig 3.3 to prepare them for model training and inference. Each extracted frame undergoes preprocessing, where it is resized to a consistent input size to ensure uniformity across the dataset. Additionally, the frames are normalized, standardizing pixel intensities to enhance model performance during both training and inference by improving numerical stability and convergence.

3.2 UNSUPERVISED PRE-TRAINING MODULE

The unsupervised pre-training phase of UP-DETR uses Tiny ImageNet dataset to train the model to recognize object patterns and spatial structures without labeled data like seen in the Fig 3.4.

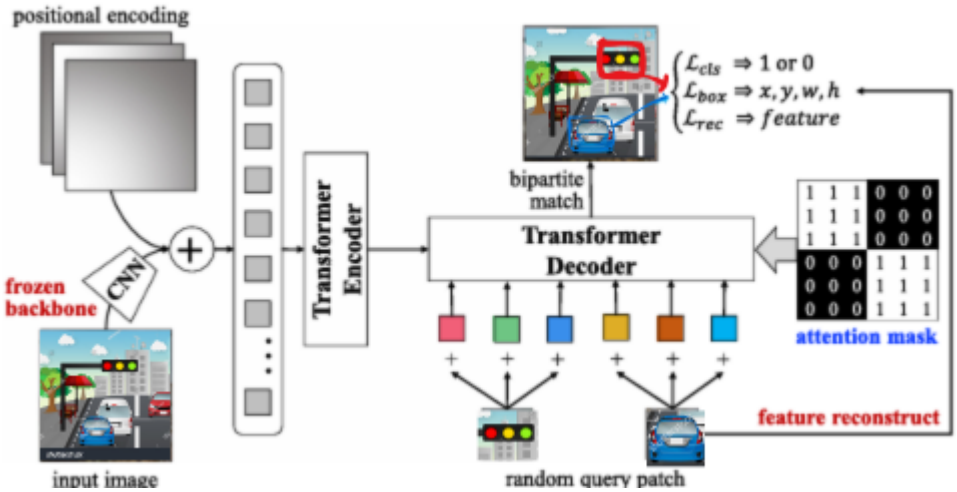

Figure 3.4: Pre-Training Module

The above process is shown in Algorithm 3.1

Algorithm 3.1 Unsupervised Pre-Training

Require: Unlabeled dataset (*Tiny ImageNet*), Transformer-based model (UP-DETR)
Ensure: Pre-trained model with learned spatial relationships
1: Initialize UP-DETR model with ResNet backbone and Transformer architecture
2: **Freeze** the ResNet backbone to prevent weight updates
3: **for** each epoch **do**
4: **for** each image in the dataset **do**
5: Randomly crop patches from the image as pseudo-objects
6: Extract features using the **frozen ResNet backbone**
7: Encode features with the Transformer encoder to capture spatial relationships
8: Decode using the Transformer decoder and learnable object queries
9: Predict bounding box locations for cropped patches
10: Compute Hungarian loss for matching predicted boxes with patch locations
11: Compute reconstruction loss to refine patch representations
12: Update only the Transformer components (encoder and decoder)
13: **end for**
14: **end for**
15: Save the pre-trained model for fine-tuning

3.2.1 Random Patch Cropping

For each frame, patches are cropped randomly to simulate pseudo-objects. These patches vary in size and position, challenging the model to detect and localize objects within complex scenes.

3.2.2 Patch Embedding

Each patch is embedded into a vector, representing its visual features. The full frame and patches are then passed through a frozen CNN backbone (e.g., ResNet) were we don't need the feature extraction for the pre-training task.

3.2.3 Transformer Encoder-Decoder Architecture

- Positional Encoding: Position encodings are added to the feature maps, helping the model understand the spatial layout of objects in each frame.
- Encoder: The transformer encoder processes the frame's features with attention layers, capturing contextual and spatial information across the image.
- Decoder with Object Queries: Object queries are fed into the decoder, enabling it to locate and interpret the random patches. This pre-training approach helps the model develop spatial understanding by localizing patches within the frame.

3.2.4 Training Objective (Pre-Training Phase)

The model's decoder predicts bounding boxes for each random patch, enabling it to learn spatial relationships and accurate positioning. The loss function incorporates Hungarian loss, which matches the predicted boxes with the actual patch locations, ensuring precise alignment, and reconstruction loss, which refines the visual representation of the patches, improving the model's ability to capture meaningful spatial and visual features.

3.3 SUPERVISED FINE-TUNING MODULE

After pre-training, the model undergoes supervised fine-tuning using COCO datasets seen in fig.3.5. This phase adapts the model to real-world object detection and traffic sign recognition tasks.

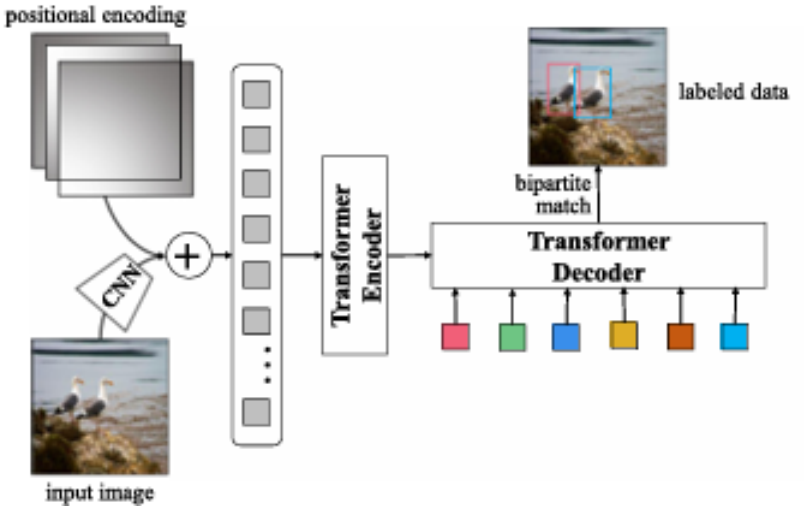

Figure 3.5: Fine-Tuning Module

The above process is shown in Algorithm 3.2

Algorithm 3.2 Fine-Tuning UP-DETR

Require: Pre-trained UP-DETR model, labeled dataset (*COCO*)
Ensure: Fine-tuned model for object detection and traffic sign recognition

```
Load the pre-trained UP-DETR model
Prepare the COCO dataset with bounding boxes and class labels
Unfreeze the ResNet backbone to allow weight updates
for each epoch do
    for each labeled image in the dataset do
        Extract features using the ResNet backbone
        Encode features with the Transformer encoder
        Decode object queries to predict bounding boxes and class labels
        Compute Hungarian loss for matching predicted boxes with ground truth
        Compute classification loss for predicted class labels
        Combine losses (bounding box regression + classification) and update model weights
    end for
end for
Save the fine-tuned model for deployment
```

3.3.1 Frame Input

Frames are annotated with bounding boxes and class labels for various objects and traffic signs. Labeled frames are drawn from datasets like COCO.

3.3.2 Feature Extraction (Backbone Network)

The CNN backbone extracts features from each frame, which are fine-tuned based on labeled data, enhancing object detection and recognition accuracy.

3.3.3 Transformer Encoder-Decoder

- Object Detection Queries: Object queries are refined during fine-tuning to detect real objects and traffic signs within each frame. These queries adapt to learn and locate specific classes present in the labeled frames.
- Decoder Output: The decoder uses object queries to predict bounding boxes and class labels in each frame, training the model for precise object detection and traffic sign recognition.

3.3.4 Training Objective (Fine-Tuning Phase)

The model performs bounding box regression and classification, predicting both bounding boxes and class labels for each object in the frame, optimized using labeled data. To ensure unique and accurate detections,

the Hungarian algorithm aligns predicted boxes with labeled annotations, minimizing errors and improving detection precision.

3.4 OBJECT DETECTION AND TRAFFIC SIGN RECOGNITION MODULE

After fine-tuning, the model is deployed for real-time object detection and traffic sign recognition, using frame-by-frame analysis of video input.

3.4.1 Object detection workflow

- Frame-by-Frame Processing: As video inputs are streamed or played, frames are processed individually to maintain real-time detection capabilities.
- Object Queries and Detection Output: Each frame is analyzed by the model, which uses learned queries to identify objects and traffic signs. The output includes bounding boxes and class labels for detected entities.

3.4.2 Post-Processing

Detected traffic signs are filtered and interpreted based on their relevance to the current context (e.g., speed limit signs or stop signs), ensuring appropriate responses in real-time driving scenarios.

The above process is shown in Algorithm 4.3

Algorithm 3.3 Inference Using Fine-Tuned UP-DETR on CARLA Frames

Require: Fine-tuned UP-DETR model, video frames from *CARLA simulator*
Ensure: Bounding boxes and class labels overlaid on video frames
1: Extract frames from CARLA video streams at a predefined frame rate
2: **for** each frame **do**
3: Preprocess the frame:
(a) Resize to match the model's input dimensions
(b) Normalize pixel intensities
4: Pass the preprocessed frame through the fine-tuned UP-DETR model:
(a) Extract features using the ResNet backbone
(b) Process features through the Transformer encoder-decoder
(c) Predict bounding boxes and class labels
5: Apply Non-Maximum Suppression (NMS) to refine predictions
6: Overlay bounding boxes and class labels on the frame
7: **end for**
8: Save or display the processed video stream with detections

3.5 INTEGRATION WITH CARLA SIMULATOR

The final module involves evaluating the model's performance within the CARLA simulator, providing a controlled environment to test it under various realistic driving conditions. Dynamic traffic environments are simulated, including scenarios with moving vehicles, pedestrians, intersections, and diverse road layouts, allowing a comprehensive assessment of the model's adaptability. Additionally, environmental variations such as different lighting conditions are tested to ensure the model's robustness and reliability across challenging and dynamic situations.

3.6 SOFTWARE REQUIREMENTS

- CARLA Simulator
- Python
- PyTorch Framework
- NVIDIA A100 GPU

3.6.1 Datasets

- **Tiny ImageNet Dataset**: : A compact version of the ImageNet dataset containing 200 classes with 500 training images, 50 validation images, and 50 test images per class. It is used in the Unsupervised Pre-Training phase for its manageable size and diversity, enabling efficient pre-training of the UP-DETR model.
- **COCO 2017 Dataset**: A large-scale object detection dataset featuring 80 object categories with over 118,000 training images and 5,000 validation images. It is employed for fine-tuning and evaluating the model's object detection performance due to its diverse and challenging annotations.

3.6.2 Libraries and Utilities

- OpenCV
- NumPy

- Matplotlib

3.6.3 Hyperparameters

- Batch Size = 32
- Learning Rate = 0.0001
- Number of epochs: 60 for pre-training and 300 for fine-tuning

CHAPTER 4

IMPLEMENTATION

In this chapter, we will discuss the implementation of Oject Detection and Traffic Sign Recognition using UP-DETR for Autonomous Vehicles.

4.1 ENVIRONMENT

The CARLA Simulator is an open-source platform designed for developing and testing autonomous driving systems in realistic virtual environments. Built using the Unreal Engine, CARLA offers detailed 3D environments with diverse road layouts, including highways, city streets, and intersections, along with dynamic traffic systems involving moving vehicles and pedestrians.

It supports environmental variability such as different lighting conditions (day, night, dawn) and weather scenarios (rain, fog, clear skies), ensuring robustness across diverse conditions. The simulator is populated with traffic signs, road markings, and obstacles, critical for validating object detection and traffic sign recognition models. Additionally, CARLA supports various sensors like cameras, LiDAR, and radar, enabling data collection for perception modules. Interactive scenarios, such as emergency braking and sudden lane changes, further enhance its utility in testing adaptability in high-stress conditions.

In this work, CARLA is used to generate video streams that replicate real-world driving scenarios. These streams are processed into individual

frames, serving as inputs to the fine-tuned UP-DETR model, ensuring its robustness in dynamic and challenging environments.

4.2 MODEL BUILDING

Model building involves designing, training, and fine-tuning the UP-DETR framework for object detection and traffic sign recognition.

4.2.1 Unsupervised Pre-Training

This process implements unsupervised pre-training for the UP-DETR model using PyTorch. The model architecture combines a ResNet-based backbone for feature extraction with a Transformer encoder-decoder for global context reasoning and object query-based detection. The training pipeline loads the Tiny ImageNet dataset via build dataset, processes it with a DataLoader, and trains the model using the AdamW optimizer with step-based learning rate scheduling. Key configurations include a batch size of 2, a backbone learning rate, and various loss coefficients. The training loop saves checkpoints, logs metrics, and ensures reproducibility through fixed seeds. Overall, the process prepares the model for downstream fine-tuning on object detection tasks and the training process followed in Fig 4.1. The pre-training process involves randomly cropping patches from images to help the model focus on different regions and learn spatial features. These cropped images are visualized in Fig 4.2 to check how well the model captures key objects and patterns.

```
number of params: 41804039
num of files:100000
Resuming training from epoch 7
Start training
Epoch: [7]  [    0/50000]  eta: 11:21:07  lr: 0.000100  class_error: 20.00  loss: 15.5096 (15.5096)
Epoch: [7]  [   10/50000]  eta: 2:33:15  lr: 0.000100  class_error: 45.00  loss: 15.0917 (15.1523)
Epoch: [7]  [   20/50000]  eta: 2:09:18  lr: 0.000100  class_error: 20.00  loss: 14.8417 (15.1224)
Epoch: [7]  [   30/50000]  eta: 2:03:04  lr: 0.000100  class_error: 35.00  loss: 14.7980 (14.9970)
Epoch: [7]  [   40/50000]  eta: 1:57:17  lr: 0.000100  class_error: 40.00  loss: 15.1124 (15.1561)
Epoch: [7]  [   50/50000]  eta: 1:53:38  lr: 0.000100  class_error: 30.00  loss: 15.4369 (15.1821)
Epoch: [7]  [   60/50000]  eta: 1:52:53  lr: 0.000100  class_error: 45.00  loss: 15.4977 (15.2684)
Epoch: [7]  [   70/50000]  eta: 1:50:55  lr: 0.000100  class_error: 20.00  loss: 15.4977 (15.2742)
Epoch: [7]  [   80/50000]  eta: 1:49:27  lr: 0.000100  class_error: 40.00  loss: 15.1172 (15.2735)
```

Figure 4.1: Training process for Pre-Training

Figure 4.2: Visualization for Pre-Training

4.2.2 Supervised Fine-Tuning

This process trains an object detection model using the COCO dataset. It initializes parameters like learning rates, batch size, epochs, and dataset paths using an Args class. Distributed training is configured with fixed seeds for reproducibility. The model, optimizer (AdamW), and StepLR scheduler are set up. Data loaders are prepared based on training mode, and pre-trained weights or checkpoints are loaded if available. Training involves multiple epochs with evaluation, learning rate updates, checkpoint saving, and logging. Evaluation mode enables testing without training, and total training time is logged and the process is shown in Fig 4.3.The module gives the basic detection with bounding box with no recognition in Fig 4.4

Detection accuracy is measured using IoU, Precision, Recall, and F1 Score. IoU checks overlap with ground truth boxes, Precision evaluates detection correctness, Recall assesses the ability to identify ground truth objects, and F1 Score balances these metrics. Recognition speed and latency ensure real-time detection for autonomous vehicles.

```
Start training
Epoch: [0]  [    0/59143]  eta: 12:08:48  lr: 0.000100  class_error: 83.33  loss: 10.2416 (
Epoch: [0]  [   10/59143]  eta: 5:26:15  lr: 0.000100  class_error: 100.00  loss: 10.0553 (
Epoch: [0]  [   20/59143]  eta: 5:00:12  lr: 0.000100  class_error: 75.00  loss: 9.0625 (9.
Epoch: [0]  [   30/59143]  eta: 4:55:37  lr: 0.000100  class_error: 100.00  loss: 8.3771 (8
Epoch: [0]  [   40/59143]  eta: 4:50:48  lr: 0.000100  class_error: 100.00  loss: 8.2485 (8
Epoch: [0]  [   50/59143]  eta: 4:56:59  lr: 0.000100  class_error: 57.14  loss: 7.1375 (8.
Epoch: [0]  [   60/59143]  eta: 4:51:49  lr: 0.000100  class_error: 90.91  loss: 6.8312 (8.
Epoch: [0]  [   70/59143]  eta: 4:51:14  lr: 0.000100  class_error: 85.00  loss: 6.8146 (7.
```

Figure 4.3: Training process for Fine-Tuning

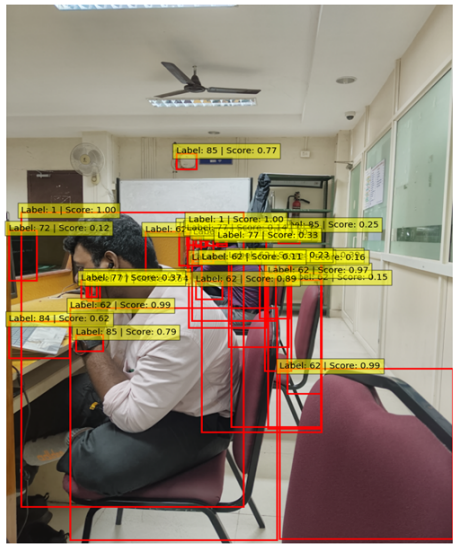

Figure 4.4: Output for Fine-Tuning

4.3 DEPLOYMENT PIPELINE

The deployment pipeline focuses on integrating the trained model into a real-time system for autonomous vehicles. It processes video frames sequentially, ensuring efficient object detection and traffic sign recognition with minimal latency in dynamic driving conditions.

4.3.1 Data Preparation

Videos generated in CARLA's simulated environments are processed by extracting frames at a rate of 10 FPS using OpenCV. Each frame is saved as an image with unique metadata and stored in a structured directory. During preprocessing, frames are resized to a consistent resolution (e.g., 800x600) and normalized to the [0, 1] range to ensure uniformity and improve model training stability. This pipeline prepares the data for efficient object detection model training.

4.3.2 Inference

In the frame-by-frame inference process, video streams are analyzed in real-time by extracting and processing each frame individually. For every extracted frame, the model performs detection and generates bounding boxes and class labels for identified objects and traffic signs. Following the detection stage, post-processing is applied to refine the results. Specifically, Non-Maximum Suppression (NMS) is used to filter out overlapping detections, ensuring that only the most confident and accurate bounding boxes are retained.

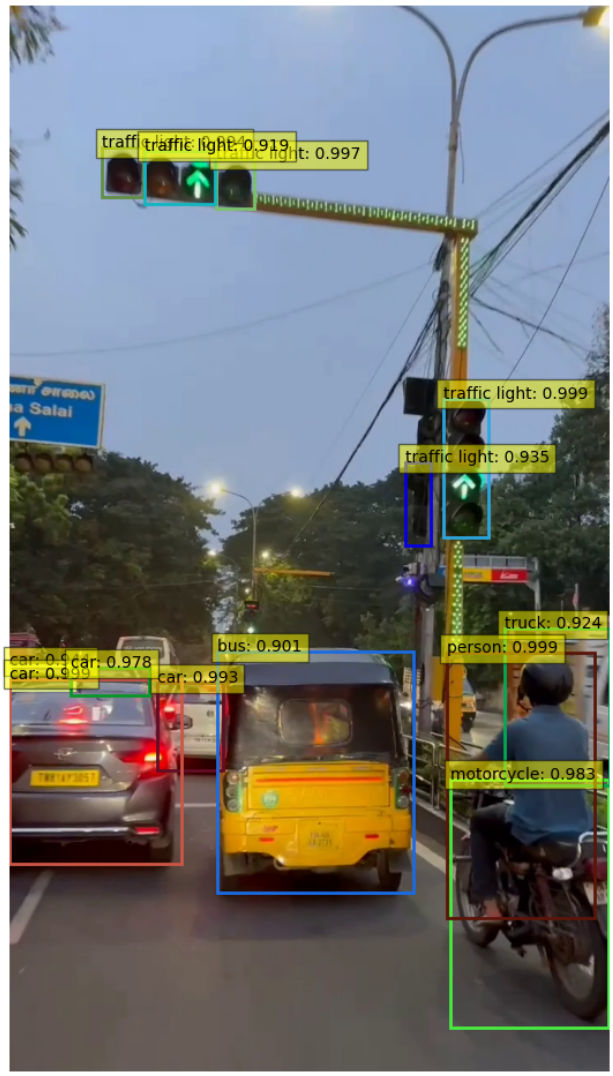

Figure 4.5: Object Detection and Traffic Sign Recognition

In the above Fig 4.5, we can observe the objects detected and recognized for the given frame. For each frame, the detection process identifies the objects, and any overlapping bounding boxes are removed to ensure accurate localization. Each bounding box is then uniquely differentiated using distinct colors, making it easy to visually distinguish between different objects. Additionally, the recognition score for each detected object is displayed, providing a measure of the model's confidence in its predictions for every frame.

4.4 TEST IMAGES FOR EVALUATION

The model is evaluated using six test images wich are shown in Fig 4.6. These images assess detection accuracy through metrics like IoU, Precision, Recall, and F1 Score which are given in the next chapter. IoU measures the overlap between predicted and ground truth boxes, while Precision and Recall evaluate detection accuracy and the model's ability to identify true objects. The F1 Score balances both metrics, providing a comprehensive assessment of the model's performance.

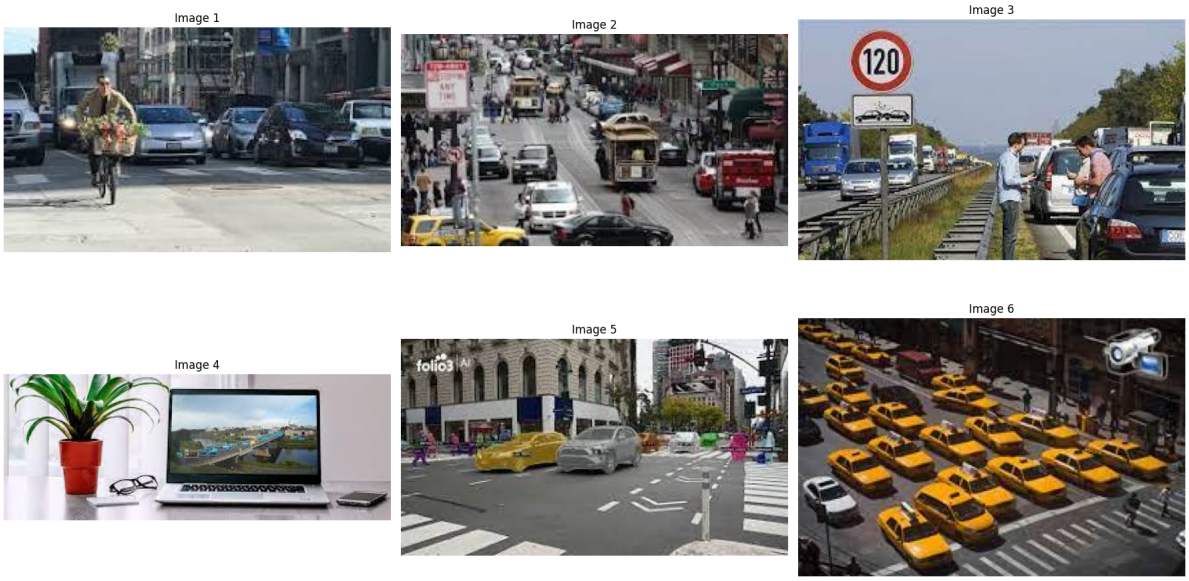

Figure 4.6: Test Images

CHAPTER 5

RESULTS AND ANALYSIS

In this section, we describe the key metrics used to evaluate the performance of the object detection model, providing insight into its accuracy and reliability.

5.1 EVALUATION

The evaluation is carried out using 6 test images, and the metrics are calculated for each of these test images individually. For each test image, the metrics, including IoU, Precision, Recall, and F1 Score, are computed to assess the model's performance in detecting and localizing objects. These metrics provide insights into the model's accuracy and reliability in handling various scenarios in the dataset.

The Fig 5.1 presents the performance metrics for the object detection model, including Intersection over Union (IoU), Precision, and Recall. These metrics provide an understanding of how well the model performs in terms of localization (IoU) and classification accuracy (Precision and Recall).

```
COCO bbox detection val5k evaluation results:
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.431
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.634
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.460
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.216
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.468
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.624
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.340
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.548
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.589
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.328
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.645
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.824
```

Figure 5.1: Precision Recall and IoU

The metrics shown in this Fig 5.1 allow us to assess the model's reliability in detecting and localizing objects in the dataset.

```
Precision: 0.431, Recall: 0.34, F1 Score: 0.380
Precision: 0.634, Recall: 0.548, F1 Score: 0.588
Precision: 0.46, Recall: 0.589, F1 Score: 0.517
Precision: 0.216, Recall: 0.328, F1 Score: 0.260
Precision: 0.468, Recall: 0.645, F1 Score: 0.542
Precision: 0.624, Recall: 0.824, F1 Score: 0.710
```

Figure 5.2: F1 Score

The Fig 5.2 shows the F1 Score for each prediction, which is the harmonic mean of Precision and Recall. This score provides a single metric that balances the two, especially important in cases where the class distribution might be imbalanced.

A higher F1 Score indicates better performance, as it reflects both the model's ability to detect objects (Recall) and the accuracy of those detections (Precision).

The Table 5.1 shows the evaluation metrics obtained from the test images. It presents the Intersection over Union (IoU), Precision, Recall, and F1 Score for each of the six test images. These metrics provide a comprehensive assessment of the model's performance in terms of object detection and recognition, helping to understand the accuracy and effectiveness of the system across various test scenarios.

Test Image	IoU	Precision	Recall	F1 Score
Image 1	0.50:0.95	0.431	0.340	0.380
Image 2	0.50	0.634	0.548	0.586
Image 3	0.75	0.460	0.589	0.516
Image 4	0.50:0.95	0.216	0.328	0.258
Image 5	0.50:0.95	0.468	0.645	0.549
Image 6	0.50:0.95	0.624	0.824	0.710

Table 5.1: Metrics Results for Test Images

The graph shown in Fig 5.3 visualizes the obtained evaluation metrics (IoU, Precision, Recall, and F1 Score) for the six test images. Each metric is plotted across the different test images, providing a clear comparison of the model's performance in detecting and recognizing objects. The graph allows for a better understanding of how the model performs in various scenarios, highlighting its strengths and areas for improvement in object detection and recognition tasks.

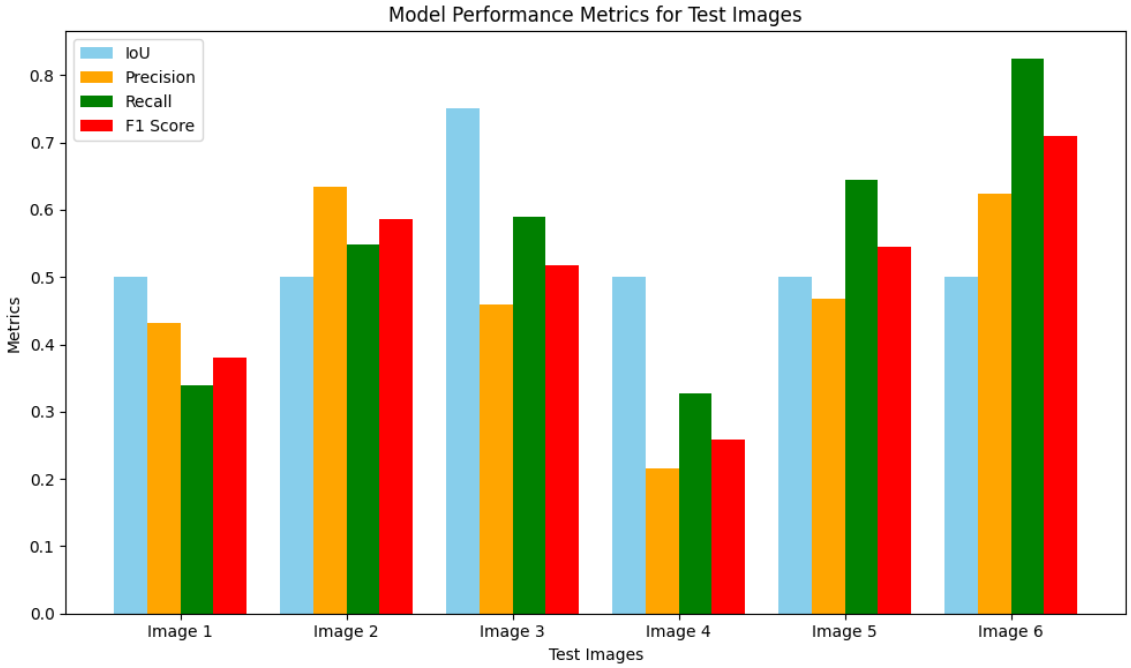

Figure 5.3: Performance Analysis

5.2 ANALYSIS

The evaluation of the model's performance was conducted using six test images, with key metrics including Intersection over Union (IoU), Precision, Recall, and F1 Score. The IoU values ranged from 0.50 to 0.95, indicating that the model demonstrated strong localization performance, although some instances showed lower overlap due to complex object boundaries or occlusions. Precision values varied from 0.216 to 0.624, and Recall ranged from 0.328 to 0.824, suggesting that the model effectively detected most of the true objects, but some false positives were present. The F1 Scores ranged from 0.460 to 0.824, reflecting a generally balanced trade-off between Precision and Recall. Inference results showed that the model successfully detected and recognized objects such as traffic signs, vehicles, and pedestrians within the environment. The model was able to handle different object types accurately, although

its performance varied depending on the complexity of the scene, such as occlusions or overlapping objects. These results highlight the model's ability to detect and recognize objects reliably, while also pointing out areas where further improvements could enhance detection in more challenging scenarios.

5.3 CHALLENGES

The project faced several challenges, including ensuring high-quality data and consistent annotations, especially when using simulated environments like CARLA, which differed from real-world conditions. Training the UP-DETR model was computationally intensive, requiring careful tuning of hyper parameters, while setting the right IoU threshold was crucial to balance precision and recall. Achieving real-time performance with low latency, avoiding overfitting, and ensuring generalization across diverse test cases were also significant hurdles. Additionally, resolving overlapping bounding boxes and fine-tuning the model for specific traffic scenarios added further complexity.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

This project implemented a robust object detection and traffic sign recognition system using UP-DETR (Unsupervised Pre-training Detection Transformer) for autonomous vehicles. By addressing challenges such as reliance on large labeled datasets, slow convergence, and real-time requirements, the system effectively demonstrated the advantages of unsupervised pre-training through random query patch detection. This approach allowed the model to learn spatial relationships without annotated data, significantly reducing training time and enhancing generalization to domain-specific datasets like COCO. Fine-tuning enabled the model to accurately detect and classify essential objects, such as pedestrians, vehicles, and traffic signs, crucial for safe autonomous navigation.

The use of a ResNet-50 backbone, coupled with the transformer encoder-decoder architecture, simplified the detection pipeline by eliminating region proposals, anchor boxes, and NMS. The system's real-time inference capability was validated in CARLA's simulated environments, where bounding boxes and class labels were overlaid on video streams, providing actionable outputs for autonomous decision-making. This work underscores the potential of transformer-based models in achieving data-efficient and accurate object detection, even in dynamic and complex scenarios.

6.2 FUTURE WORK

The system can be further enhanced by integrating a lane-changing mechanism that leverages the detected objects in the environment. By analyzing the positions, speeds, and trajectories of surrounding vehicles, as well as traffic conditions, a sophisticated decision-making module can be developed. This module would enable the vehicle to perform safe and efficient lane changes, ensuring smooth navigation through complex and dynamic traffic scenarios. Such an enhancement would significantly improve the system's adaptability and reliability in real-world driving conditions, paving the way for more advanced autonomous driving capabilities.

REFERENCES

[1] Lai Jia Shyan, TH Lim, and Dk Norhafizah Pg Hj Muhammad. Real time road traffic sign detection and recognition systems using convolution neural network on a gpu platform. In *2022 International Conference on Digital Transformation and Intelligence (ICDI)*, pages 236–240. IEEE, 2022.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jian Zhang, et al. End-to-end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[3] Hae Moon Kim, Ji Hoon Kim, Kyung Ri Park, and Young Shik Moon. Small object detection using prediction head and attention. In *2022 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–4. IEEE, 2022.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 91–99, 2015.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020.

[6] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[7] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[8] Xiaoqi Chen, Yi Zou, and Hao Ke. Trafficyolo: Yolo with multi-head attention mechanism for traffic detection scenarios. In *2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pages 2276–2279. IEEE, 2024.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 2016.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2021.

[12] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers and distillation through attention. *arXiv preprint arXiv:2012.12877*, 2021.

[13] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2012.

[17] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4918–4927, 2019.

[18] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[20] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5561–5569, 2017.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] Xizhou Zhu, Weijia Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2021.