Dynamic Detection and Segmentation: Submitted to IIT TIRUPATI

submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

M SHILPA CS21B034

C DAKSHAYANI CS21B016

Supervisor(s)

Dr. Vishnu Chalavadi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

May 2025

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. we also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission to the best of my knowledge. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Place: Tirupati
Date: 11-05-2025

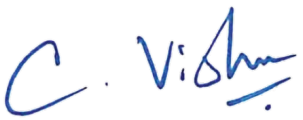
Signature
M shilpa
CS21B034

Signature
C Dakshayani
CS21B016

Note: If more than one authors mentioned in the cover page, modify each 'I' by 'we' and then include remaining author's name at the bottom of the page. Signature of each author in this page is must.

BONA FIDE CERTIFICATE

This is to certify that the report titled **Dynamic Detection and Segmentation** , submitted by **M shilpa, C Dakshayani**, to the Indian Institute of Technology, Tirupati, for the award of the degree, is a bona fide record of the project work done by them under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.



Place: Tirupati
Date: 11-05-2025

Dr. Vishnu Chalavadi
Guide
Assistant Professor
Department of Computer Science and Engineering
IIT Tirupati - 517501

Note: If supervised by more than one professor, professor's name must be included and get signatures from all supervisors. Change him/her or our or my accordingly.

ACKNOWLEDGMENTS

We would like to thank our guide Dr. chalavadi vishnu for guiding and supporting us throughout this project. Also we are greatful to Facebook AI for developing the DETR (Detection Transformer),and Mask2former (Segmentation Transformer)which served as the foundation for our work on Dynamic Detection and Segmentation.

ABSTRACT

This project presents a dynamic detection and segmentation framework that allows selective processing of objects based on their importance in a given context. The system is trained on two classes: pedestrians and cars. Unlike conventional models that perform both detection and segmentation for all classes, our approach enables users to specify which objects should be segmented and which should only be detected. For example, if precise size estimation is critical for pedestrians, segmentation is applied, while less critical objects like cars can be limited to detection only. This flexible strategy reduces computational overhead and memory usage by avoiding unnecessary segmentation, making the system more efficient and adaptable to real-world scenarios.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

DETR	Detection Transformer
IOU	Intersection over Union
AP	Average Precision
AR	Average Recall
FP	False Positive
FN	False Negative
TP	True Positive
mAP	Mean Average Precision
mAR	Mean Average Recall
NMS	Non-Max Suppression
LR	Learning Rate
GPU	Graphics Processing Unit
CNN	Convolutional Neural Network
Bbox	Bounding Box
Epoch	Training Cycle
MSE	Mean Squared Error
ResNet	Residual Network
ReLU	Rectified Linear Uni
RAM	Random Access Memory

NOTATION

F'	Feature map after positional encoding, $F' = F + P$
Q	Query matrix, $Q = F'W_Q$
K	Key matrix, $K = F'W_K$
V	Value matrix, $V = F'W_V$
d_k	Dimension of key vectors
Attention	$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$
$\hat{b}_i$	Predicted bounding box for query i, $\hat{b}_i \in \mathbb{R}^4$
z_i	Output embedding of decoder for query i
$W_{\textbf{bbox}}$	Weight matrix for bounding box prediction
$b_{\textbf{bbox}}$	Bias vector for bounding box prediction
$p_{\textbf{class},i}$	Predicted class probabilities for object i
$W_{\textbf{class}}$	Weight matrix for class prediction
$b_{\textbf{class}}$	Bias vector for class prediction
$\mathscr{L}_{\textbf{cls}}$	Classification loss, $\mathscr{L}_{\text{cls}} = -\sum_i y_i \log(p_{\text{class},i})$
$\mathscr{L}_{\textbf{bbox}}$	Bounding box loss, $\mathscr{L}_{\text{bbox}} = \sum_i \|\hat{b}_i - b_i\|_1$
M_i	Predicted segmentation mask for object i, $M_i = \sigma(W_{\text{mask}} \cdot z_i + b_{\text{mask}})$
$W_{\textbf{mask}}$	Weight matrix for mask prediction
$b_{\textbf{mask}}$	Bias vector for mask prediction
σ	Sigmoid activation function
$\mathscr{L}_{\textbf{mask}}$	Binary cross-entropy loss for masks, $\mathscr{L}_{\text{mask}} = -\sum_i [y_i \log M_i + (1-y_i)\log(1-M_i)]$
$\mathscr{L}_{\textbf{total}}$	Total loss, $\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{cls}} + \lambda_{\text{bbox}}\mathscr{L}_{\text{bbox}} + \lambda_{\text{mask}}\mathscr{L}_{\text{mask}}$
AP	Average Precision, $AP = \int_0^1 \text{Precision}(\text{Recall})\, d(\text{Recall})$
Precision	Precision $= \frac{TP}{TP+FP}$
Recall	Recall $= \frac{TP}{TP+FN}$

CHAPTER 1

INTRODUCTION

In many computer vision applications, it is not always necessary to perform both object detection and segmentation for every object in an image. Detection provides object location through bounding boxes, while segmentation gives detailed pixel-level information. However, segmentation is computationally more expensive. This project introduces a dynamic system that allows selective application of detection and segmentation depending on the importance of each object class.

The model is trained on two classes: pedestrians and cars. For example, if accurate size and shape information is required for pedestrians, segmentation is applied. On the other hand, if cars are less critical in a given scenario, detection alone is used. This selective approach helps reduce memory usage and computational load, making the system more efficient and adaptable to real-time use cases.

1.1 Objectives and Scope

The objective of this project is to develop a smart and efficient system that can perform object detection and segmentation selectively, depending on the importance of each object in a given context. Unlike traditional models that apply both tasks to all objects, this system allows users to choose which object classes should be segmented and which should be simply detected. This helps reduce unnecessary computation and memory usage, especially in real-time or resource-limited environments.

The project focuses on two object classes: pedestrians and cars. For situations where accurate shape and size are needed, such as tracking people for safety purposes, segmentation can be applied to pedestrians. In contrast, if cars are not the main focus, the system will only detect them using bounding boxes. This approach keeps the process lightweight without sacrificing accuracy where it truly matters. The model is designed to be flexible and can be extended to include more classes in future use cases, making it suitable for applications like traffic analysis, surveillance, and smart city systems.

CHAPTER 2

Literature Review

The integration of object detection and semantic segmentation has evolved rapidly, driven by the need for more detailed and flexible scene understanding in real-world applications such as autonomous driving and surveillance. Traditionally, object detection models such as Faster R-CNN, YOLO, and SSD provide bounding boxes for object localization, while segmentation models like FCN, DeepLab, and U-Net assign class labels to each pixel. However, these tasks have often been handled separately or through joint multitask models.

In a similar direction, the OpenImage2019 challenge presented high-performing solutions that avoided single multitask models. The winning method used separate models for detection and segmentation, which were later merged intelligently. This allowed each model to be fine-tuned for its specific task, achieving better results than unified models. The key insight was that task specialization often leads to better performance than combining multiple objectives in a single network.

More recently, transformer-based models such as DETR (DEtection TRansformer) revolutionized detection by removing hand-designed components like anchor boxes and NMS (Non-Maximum Suppression), instead using an end-to-end transformer framework with set prediction. MaskFormer and Mask2Former extended this concept to segmentation, where each transformer query is responsible for both object classification and mask prediction. These models handle multiple segmentation types: semi-semantic, instance, and panoptic, under a unified architecture.

While these models offer powerful performance, they typically process all classes uniformly, without the ability to prioritize or selectively segment specific object types. Our project introduces an important innovation by enabling dynamic selection allowing users to choose which classes should be segmented (e.g., pedestrians for precise spatial measurement) and which should only be detected (e.g., cars to save memory and computation). This flexible and application-driven design has practical value, particularly in

resource constrained environments where computational cost and processing time are critical factors.

Thus, our work builds on the strengths of transformer-based vision models, while addressing a real-world gap: the need for customizable, class wise control over detection and segmentation processes.

CHAPTER 3

Dynamic Detection and Segmentation

3.1 About this Chapter

This chapter outlines the technical foundation, experimental setup, and methodologies used for implementing transformer-based models for object detection and semantic segmentation on the CamVid dataset. The dataset contains 32 semantic classes, but for this work, we focus on two specific classes: pedestrian and car, for the tasks of detection and segmentation. The user can dynamically select which class to detect and which class to segment, allowing for a flexible and task-oriented approach.

- This chapter outlines the technical foundation, experimental setup, and methodologies used for implementing transformer-based models for object detection and semantic segmentation in urban driving scenes. Specifically, this work employs the CamVid dataset, selecting pedestrian and car classes to demonstrate the effectiveness of using DETR for object detection and Mask2Former for instance segmentation.
- The chapter is structured to first explain the mathematical formulations behind the two models, providing insight into their transformer architecture, query-based design, and loss functions. Then it details the experimental procedures followed, ranging from pre-processing and model configuration to training protocols and evaluation metrics.
- Additionally, the chapter discusses memory optimization techniques and strategies applied to improve efficiency without compromising accuracy. The methodology reflects a hybrid approach where detection or segmentation is dynamically selected for each object class based on task relevance and computational overhead
- The insights, derivations, and representative metrics discussed in this chapter serve as the core foundation of this project and inform the analysis presented in subsequent sections.

3.2 Mathematical Formulation

- **Understanding DETR: Detection Transformer:**
  - **Input Image Preprocessing:**
    - The input image $I \in \mathbb{R}^{H \times W \times 3}$ is passed through a CNN backbone (e.g., ResNet) to extract feature maps. The backbone generates a feature tensor $\mathbf{F} \in \mathbb{R}^{H' \times W' \times C}$, where H' and W' are the spatial dimensions after convolution and C is the number of channels.

* Positional encodings $\mathbf{P}$ are added to the feature map $\mathbf{F}$ to preserve spatial information since transformers do not have inductive biases related to spatial structure:

$$\mathbf{F}' = \mathbf{F} + \mathbf{P}$$

where $\mathbf{P} \in \mathbb{R}^{H' \times W' \times C}$.

– **Transformer Encoder:**

* The positional encoding-augmented feature map $\mathbf{F}'$ is fed into the transformer encoder. The encoder is a stack of multi-head self-attention layers, which computes the attention between different patches of the image to learn global dependencies.
* The attention mechanism is computed as follows, where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are query, key, and value matrices respectively:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

* The output of the attention layer is passed through feed-forward neural networks and layer normalization to learn spatial dependencies across the image patches.

– **Transformer Decoder:**

* The transformer decoder takes object queries $\mathbf{q}_i \in \mathbb{R}^{1 \times C}$ and interacts with the output of the encoder. Each query corresponds to a potential object in the image. The decoder computes the updated queries as follows:

$$\mathbf{z}_i = \text{Decoder}(\mathbf{q}_i, \mathbf{F}')$$

where $\mathbf{z}_i \in \mathbb{R}^{1 \times C}$ is the decoder's output for each query.
* The decoder outputs two sets of predictions: the bounding box $\hat{b}_i$ and the object class $p_{\text{class},i}$.

– **Bounding Box Prediction:**

* The bounding boxes $\hat{b}_i \in \mathbb{R}^4$ for each query are predicted using a simple fully connected layer:

$$\hat{b}_i = W_{\text{bbox}} \cdot \mathbf{z}_i + b_{\text{bbox}}$$

where W_{bbox} and b_{bbox} are the learned weights and bias for bounding box prediction, and the output $\hat{b}_i$ contains four values representing the coordinates (x, y, width, height).

– **Class Prediction:**

* The predicted object class $p_{\text{class},i}$ is obtained using a softmax activation on the decoder's output:

$$p_{\text{class},i} = \text{softmax}(W_{\text{class}} \cdot \mathbf{z}_i + b_{\text{class}})$$

where W_{class} and b_{class} are the learned parameters for class prediction, and the softmax function ensures that the predicted values sum to 1 across all possible classes.

– **Loss Function:**

* The total loss for DETR is a combination of the classification loss,

bounding box loss, and auxiliary losses for matching predictions to ground truth.

* The classification loss L_{cls} is computed using cross-entropy:

$$L_{\text{cls}} = -\sum_i y_i \log(p_{\text{class},i})$$

where y_i is the true class for object i.

* The bounding box loss L_{bbox} is the sum of the L_1 loss between the predicted bounding box $\hat{b}_i$ and the ground truth b_i:

$$L_{\text{bbox}} = \sum_i \|\hat{b}_i - b_i\|_1$$

* The final total loss is:

$$L_{\text{total}} = L_{\text{cls}} + \lambda_{\text{bbox}} L_{\text{bbox}}$$

- **Understanding Mask2Former: Instance Segmentation with Transformers:**
  - **Input Image Preprocessing:**
    * Like DETR, the input image I is passed through a CNN backbone to extract feature maps $\mathbf{F}$, which are augmented with positional encodings $\mathbf{P}$ to preserve spatial relations:

$$\mathbf{F}' = \mathbf{F} + \mathbf{P}$$

  - **Transformer Encoder:**
    * The transformer encoder in Mask2Former works the same way as in DETR, where self-attention is used to capture dependencies between different regions of the image.
    * The encoder output $\mathbf{F}'$ is processed to learn global contextual information for each pixel.
  - **Transformer Decoder with Mask Prediction:**
    * In Mask2Former, each query corresponds to both an object and its segmentation mask. The decoder produces both the bounding box coordinates and the segmentation mask.
    * The mask prediction M_i for each object is generated using a sigmoid activation function:

$$M_i = \sigma(W_{\text{mask}} \cdot \mathbf{z}_i + b_{\text{mask}})$$

where σ is the sigmoid function, which outputs values between 0 and 1 for each pixel in the predicted mask.

  - **Bounding Box Prediction:**
    * The bounding box $\hat{b}_i$ is predicted similarly to DETR using the decoder output:

$$\hat{b}_i = W_{\text{bbox}} \cdot \mathbf{z}_i + b_{\text{bbox}}$$

  - **Class Prediction:**

* The class prediction $p_{\text{class},i}$ is also generated using a softmax layer:

$$p_{\text{class},i} = \text{softmax}(W_{\text{class}} \cdot \mathbf{z}_i + b_{\text{class}})$$

– **Loss Function:**

* The total loss for Mask2Former is a weighted sum of three components: classification loss, bounding box regression loss, and segmentation mask loss.
* The segmentation mask loss is computed using binary cross-entropy:

$$L_{\text{mask}} = -\sum_i [y_i \log(M_i) + (1 - y_i) \log(1 - M_i)]$$

where y_i is the true binary mask and M_i is the predicted mask.
* The final total loss for Mask2Former is:

$$L_{\text{total}} = L_{\text{cls}} + \lambda_{\text{bbox}} L_{\text{bbox}} + \lambda_{\text{mask}} L_{\text{mask}}$$

- **Average Precision and Average Recall:**

– **Average Precision (AP):**

* Average Precision (AP) is a common metric used to evaluate the performance of object detection models. It calculates the precision at different recall levels by integrating the precision-recall curve.
* The precision at a given recall level is computed as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

where TP is the number of true positives and FP is the number of false positives.
* The average precision is the area under the precision-recall curve, often calculated as:

$$\text{AP} = \int_0^1 \text{Precision}(\text{Recall})\, d(\text{Recall})$$

which integrates precision over all recall levels.

– **Average Recall (AR):**

* Average Recall (AR) is another important metric, which evaluates how well the model detects all relevant objects across different recall levels.
* Recall at a given precision level is calculated as:

$$\text{Recall} = \frac{TP}{TP + FN}$$

where FN is the number of false negatives.
* The average recall is the mean recall over a range of thresholds and is often reported as the mean of recalls at different intersection-over-union (IoU) thresholds.

* The average recall is given by:

$$\text{AR} = \frac{1}{N} \sum_{i=1}^{N} \text{Recall}_i$$

where N is the number of IoU thresholds considered, and Recall_i is the recall at each threshold.

- **Memory Optimization:**
  - **Memory Usage in Dynamic Detection and Segmentation:**
    * In our project, we implement a dynamic system where we can choose whether to detect or segment each object class (e.g., pedestrian or car) based on its importance. For instance, we may segment pedestrians for precise shape and size estimation, while opting to detect cars to save resources.
    * We compared the memory usage of object detection (using DETR) and instance segmentation (using Mask2Former) to evaluate the cost of each operation.
    * Instead of recording memory before and after the task, we directly measured the peak GPU memory consumption for detection and segmentation tasks under identical input conditions.
    * Our results showed that segmentation consistently required significantly more memory than detection. This is due to the added complexity of pixel-level mask computation in segmentation compared to bounding box generation in detection.
    * Based on this insight, our system dynamically applies segmentation only when necessary, reducing memory usage without compromising the precision of critical object classes.
  - **Memory Optimization Techniques:**
    * **Model Pruning:** Redundant parameters were removed from both DETR and Mask2Former models to reduce size and memory requirements while preserving task-specific accuracy.
    * **Quantization:** Reducing parameter precision (e.g., 32-bit to 16-bit) lowered memory consumption with minimal performance tradeoff.
    * **Dynamic Model Switching:** Our pipeline selects between detection and segmentation models per object class, ensuring memory-intensive segmentation is used only when needed.
    * **Batch Size Adjustment:** The batch size was tuned to fit within available memory, especially for high-resolution inputs during segmentation.
  - **Memory-Efficient Training Strategies:**
    * **Gradient Checkpointing:** Used during training to reduce memory usage by selectively storing intermediate values needed for backpropagation.
    * **Mixed Precision Training:** Leveraging lower-precision formats during training helped reduce GPU memory usage and increase throughput without accuracy loss.
  - **Memory Usage Evaluation:**
    * We evaluated memory usage by recording peak GPU memory consumption separately for the detection and segmentation tasks using identical

inputs.
  * DETR (detection) required significantly less memory compared to Mask2Former (segmentation), validating our decision to use segmentation only for objects requiring fine-grained analysis.
  * This comparison helped us optimize the pipeline to be both memory-efficient and adaptable to the task needs.

3.3 Procedure, Techniques and Methodologies

Overview

This project introduces a dynamic vision system that intelligently switches between object detection and semantic segmentation based on the relevance of the object class. For classes like **Pedestrian**, where detailed spatial understanding is crucial, semantic segmentation is employed. For less critical classes like **Car**, object detection suffices. This selective approach minimizes memory usage while maintaining high accuracy for important classes.

1. Data Preparation

The dataset includes images labeled with two primary object classes: **Pedestrian** and **Car**.

Annotations are provided in:

- Bounding box format for object detection.
- Pixel-level mask format for semantic segmentation.

The dataset is divided into training, validation, and test subsets (typically in a 70-15-15 ratio).

2. Model Selection

- **Object Detection Model**: **DETR (DEtection TRansformer)** is used for predicting bounding boxes and class labels. It is utilized for detecting object classes that don't require pixel-level accuracy .
- **Semantic Segmentation Model**: A Transformer-based semantic segmentation model such as **Mask2Former** is used to segment the class at the pixel level. This model ensures precise object delineation in cases where spatial accuracy is crucial.

3. Dynamic Task Execution Logic

At inference time, the system dynamically decides which model to use for each class in the image:

- If the object class is important when precise size is needed, the **Mask2Former** semantic segmentation model is used to segment the object at the pixel level.
- If the object class is not that important, the **DETR** object detection model is used to predict the bounding box and class label.

This dynamic task selection optimizes memory usage and inference speed while ensuring high accuracy where spatial detail is most important.

4. Training Strategy

- **Model Training**: The models were trained independently, with each model handling its respective annotation format:
  - **DETR** was trained on bounding box annotations.
  - **Mask2Former** was trained on pixel-level segmentation masks.
- **Optimization**: Batch sizes and learning rates were carefully tuned not only to optimize GPU memory usage, especially for the **Mask2Former** segmentation model, but also to ensure stable training and effective reduction of the loss throughout the learning process.

5. Evaluation and Analysis

- **Memory Usage Comparison**:
  - **Semantic segmentation** (with **Mask2Former**) requires significantly more GPU memory than object detection due to the pixel-level processing.
  - **Object detection** (with **DETR**) is more memory-efficient, making it ideal for less-detailed classes.
- **Accuracy Assessment**:
  - **mAP (mean Average Precision)** was used to evaluate the detection model's performance for object localization and classification.
  - **mIoU (mean Intersection over Union)** was used to assess the segmentation model's quality.

6. Inference Pipeline

The final system operates as follows during inference:

1. For each object class in the image, the system decides whether to apply **DETR** for detection or **Mask2Former** for segmentation based on the class:
2. The results from the detection and segmentation models are merged, and the final output is generated, ensuring efficient memory usage and fast processing without compromising accuracy.

3.3.1 Pipeline

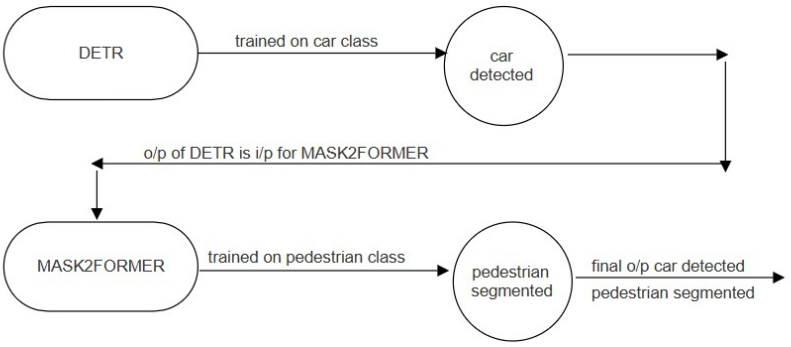


Figure 3.1: Pipeline

3.3.2 Pipeline

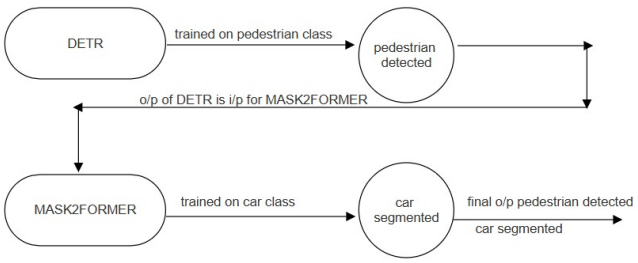


Figure 3.2: Pipeline

CHAPTER 4

Results and Discussions

4.1 Overview of the Results

This chapter presents the experimental outcomes of applying transformer-based models for object detection and semantic segmentation on the CamVid dataset. The experiments focus on two specific object classes: **pedestrian** and **car**. Each class was evaluated independently using both DETR for object detection and Mask2Former for instance segmentation.

The system is designed to dynamically allow the user to choose which class to detect and which to segment, enabling flexibility and task-based optimization. The evaluation was based on **training loss**, **precision**, and **recall** for each model-class combination.

4.2 Training Loss Analysis

Training loss was monitored across epochs to assess the convergence behavior of both models. Separate models were trained for each task (detection or segmentation) and each class (pedestrian or car).

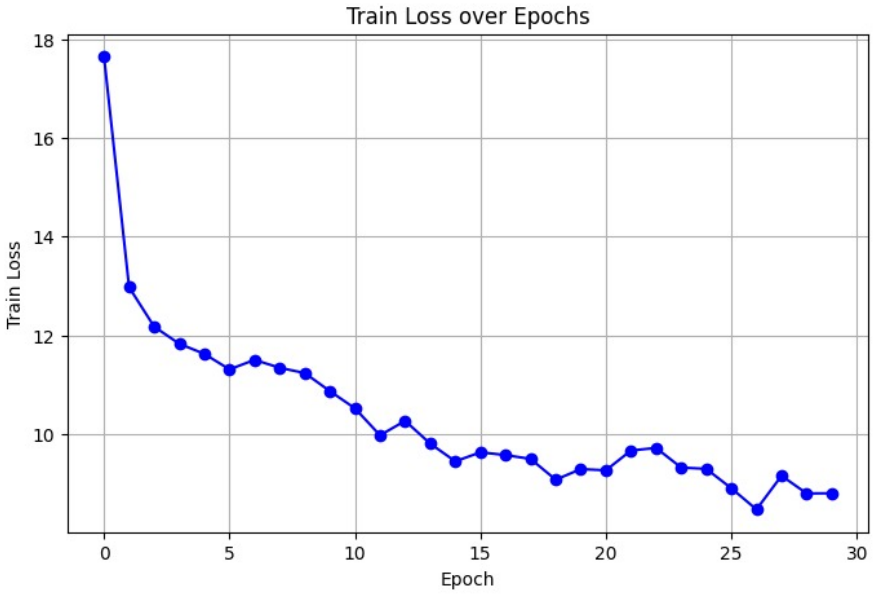


Figure 4.1: DETR Training Loss for Pedestrian Class

4.2.1 DETR Training Loss

4.2.2 Mask2Former Training Loss

4.2.3 Precision for DETR

4.2.4 Recall for DETR

4.2.5 Precision for Mask2former

4.2.6 Recall for Mask2former

4.2.7 Memory usage comparision

4.3 Qualitative Results

4.3.1 Result-1

4.3.2 Result-2

4.3.3 Result-3

4.4 Discussion

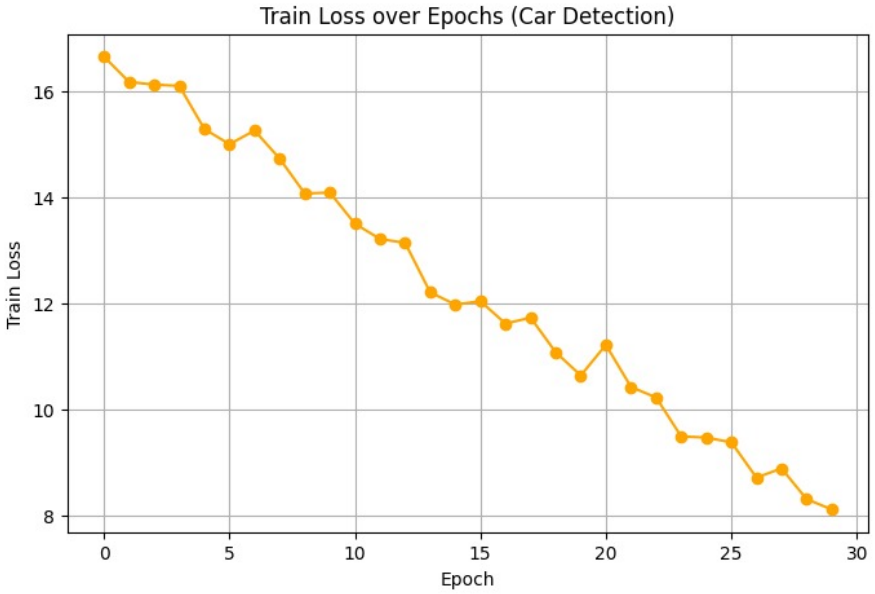


Figure 4.2: DETR Training Loss for Car Class

scenes.

- **Figure 4.1:** The DETR training loss for the pedestrian class started at about 17.5 and dropped to around 9.0 after 30 epochs. The decrease was slower and less smooth compared to the car class. This shows that the model found it harder to learn pedestrian features, likely because pedestrians are smaller and appear in different shapes.
- **Figure 4.2:** The DETR training loss for the car class started at around 16.3 and dropped to about 8.1 after 30 epochs. The loss decreased steadily, showing that the model was learning well. This smooth drop means the training was stable. Cars are easier to detect because they are bigger and have a clear shape, which helped the model train faster.
- **Figure 4.3:** The training loss for the pedestrian class using Mask2Former started at around 45 and dropped to about 22 over 30 epochs. The validation loss stayed higher, around 35, and didn't drop as much. This means the model learned well during training but found it harder to perform on new data. This is likely because pedestrians are small and look different in each image, making them harder to segment.
- **Figure 4.4:**The training loss for the car class using Mask2Former started at about 36 and dropped to around 17 by the end of 30 epochs. The validation loss stayed higher, around 30, but remained stable. This shows the model trained well and also performed consistently on new data. Cars are easier to segment because they are larger and have clear shapes
- **Figure 4.5:**The graph shows that precision starts at 0.20, dips to 0.17, and then gradually improves with fluctuations, reaching a peak of 0.31 at epoch 27. This indicates the model's performance improves over time despite some ups and downs.
- **Figure 4.6:**This graph shows the DETR model's precision (AP@[.5:.95]) for the Car class over 30 epochs. The precision starts around 0.21, gradually increases with some fluctuations, and reaches about 0.33 at epoch 30. The upward trend indicates that the model is learning well and improving its ability to detect cars more accurately over time.

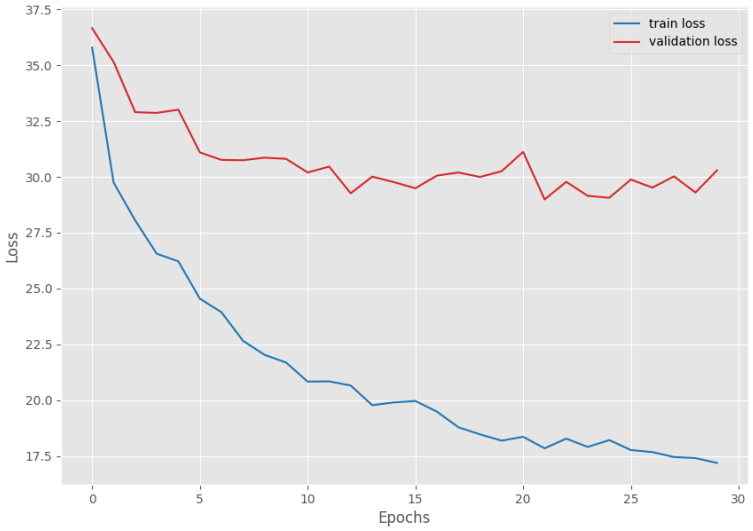


Figure 4.3: Mask2Former Training Loss for Pedestrian Class

- **Figure 4.7:**This graph shows the recall for the pedestrian class using the DETR model over 30 epochs. It starts at 0.44, drops to 0.33, but gradually improves with some ups and downs. By epoch 28, recall reaches a high of 0.68, showing that the model gets better at detecting pedestrians over time.
- **Figure 4.8:**This graph shows the recall (AR@100) for the car class using the DETR model over 30 epochs. It starts at around 0.38 and steadily increases with minor fluctuations. By the end of training, it reaches about 0.70, showing that the model becomes much better at detecting cars as training progresses.
- **Figure 4.9:**The graph shows the Mask2Former model's precision for the pedestrian class over 30 epochs. The training precision stays stable, ranging from 0.317 to 0.322, with small fluctuations. The validation precision is consistently higher, between 0.327 and 0.335, suggesting the model performs slightly better on validation data than on training data. Both precision values remain stable throughout the training.
- **Figure 4.10:**This graph shows the Mask2Former model's precision for the car class over 30 epochs. The blue line represents the training precision, which fluctuates between 0.310 and 0.317. The red line shows the validation precision, which is consistently higher, ranging from 0.320 to 0.330. The model performs slightly better on validation data compared to training data, and both precision values maintain a stable trend throughout the epochs.
- **Figure 4.11:**This graph shows the Mask2Former model's recall for the pedestrian class over 30 epochs.The recall for the pedestrian class rises quickly, then stabilizes around 0.31–0.33. Validation recall is often higher than training, suggesting good generalization. Some late fluctuations may indicate overfitting or data issues
- **Figure 4.12:**The graph shows recall for the car class using Mask2Former, increasing from 0.24 to around 0.34 over 30 epochs. Validation recall stays above training recall, suggesting the model generalizes well and may be learning more stable features for cars. Slight fluctuations in training recall may be due to class imbalance or varying car sizes in images.
- **Figure 4.13:**This graph shows GPU memory usage per image for the pedestrian

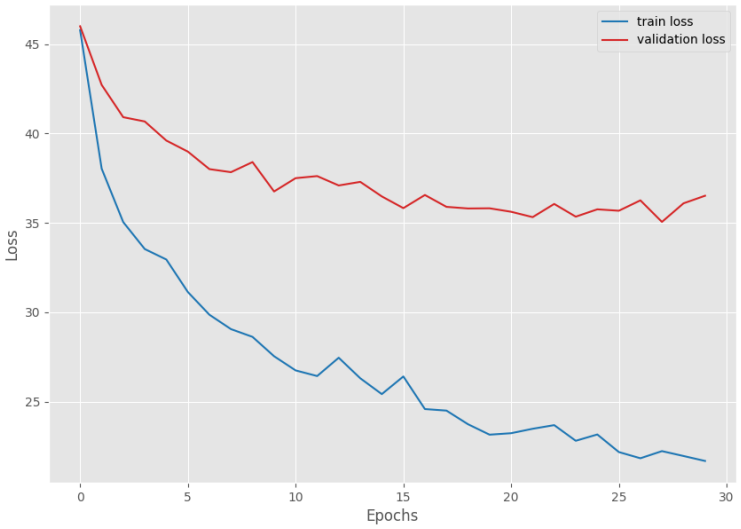


Figure 4.4: Mask2Former Training Loss for Car Class

class using DETR (detection) and Mask2Former (segmentation). Mask2Former consistently uses more memory, ranging between 3600 MB and 4200 MB, while DETR stays lower, around 2800 MB to 3200 MB. The higher memory usage in Mask2Former is expected, as segmentation requires pixel-level predictions, making it more computationally intensive than the bounding-box-based detection performed by DETR.

- **Figure 4.14:**Similar to the pedestrian class, this graph shows memory usage for the car class. Again, Mask2Former uses significantly more memory—typically between 3700 MB and 4200 MB, while DETR ranges from 2800 MB to 3300 MB. The pattern is consistent with the earlier graph: segmentation demands more GPU resources than detection, especially for larger or more detailed objects like cars.
- **Figure 4.15:**This graph shows the Mask2Former model's precision for the car class over 30 epochs. The blue line represents the training precision, which fluctuates between 0.310 and 0.317. The red line shows the validation precision, which is consistently higher, ranging from 0.320 to 0.330. The model performs slightly better on validation data compared to training data, and both precision values maintain a stable trend throughout the epochs.
- **Figure 4.15,4.16,4.17:**The first image(4.15)prompts the user to select a class (in this case, a pedestrian), and the second image(4.16) shows the result with the pedestrian class selected and the car being segmented. The third image(4.17) appears to show both the pedestrian and the car being detected and segmented.

These findings validate the flexibility and effectiveness of using DETR and Mask2Former together, with dynamic task selection based on class-specific needs.

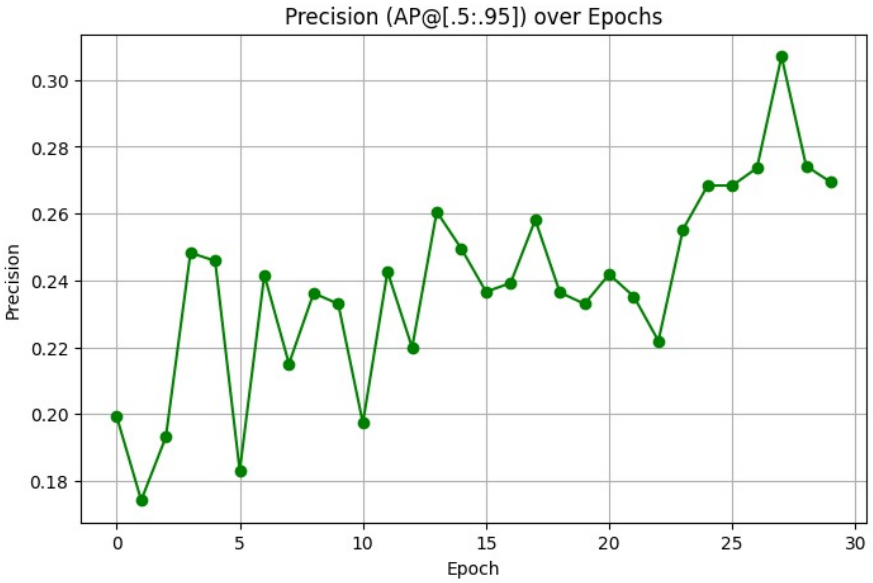


Figure 4.5: Precision using DETR for pedestrain class

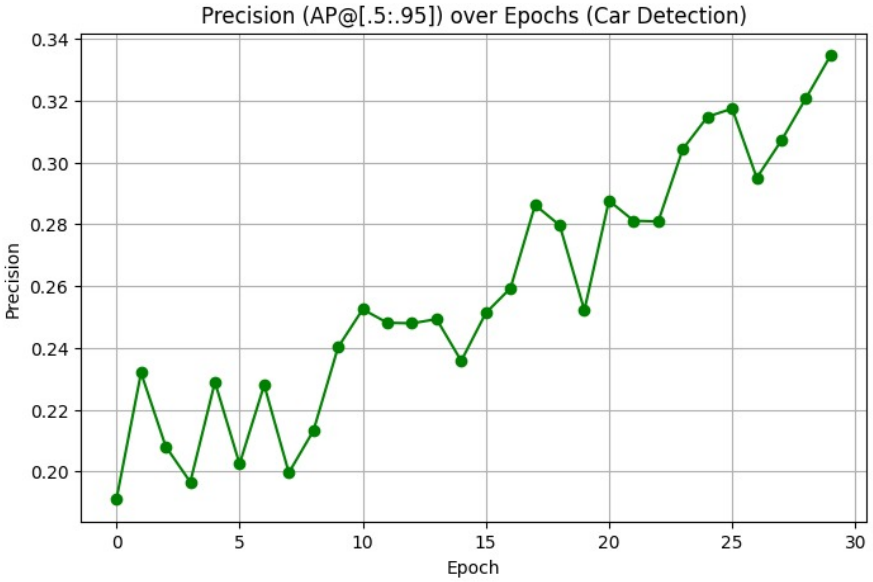


Figure 4.6: Precision using DETR for car class

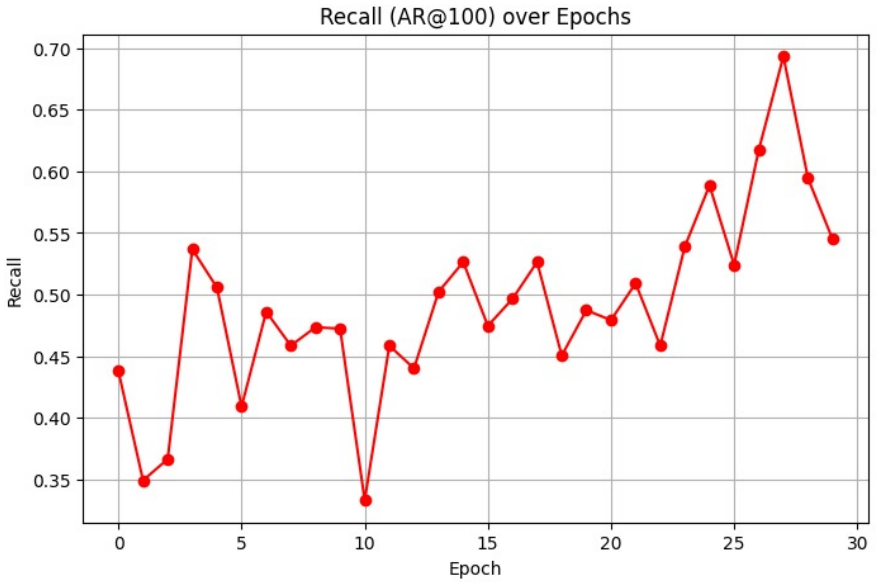


Figure 4.7: Recall using DETR for pedestrain class

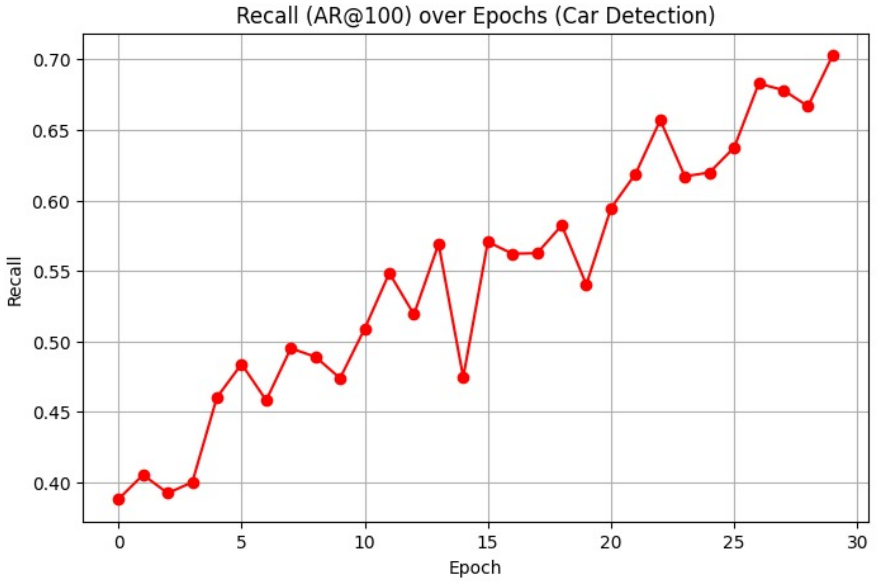


Figure 4.8: Recall using DETR for car class

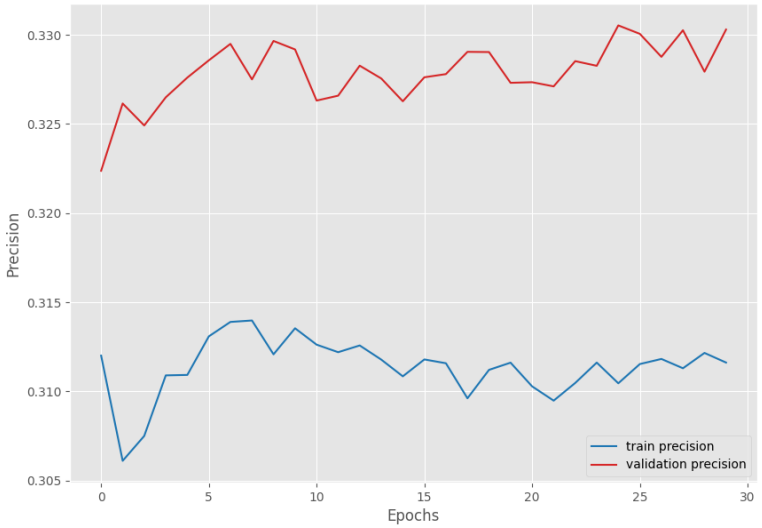


Figure 4.9: Precision using Mask2former for pedestrain class

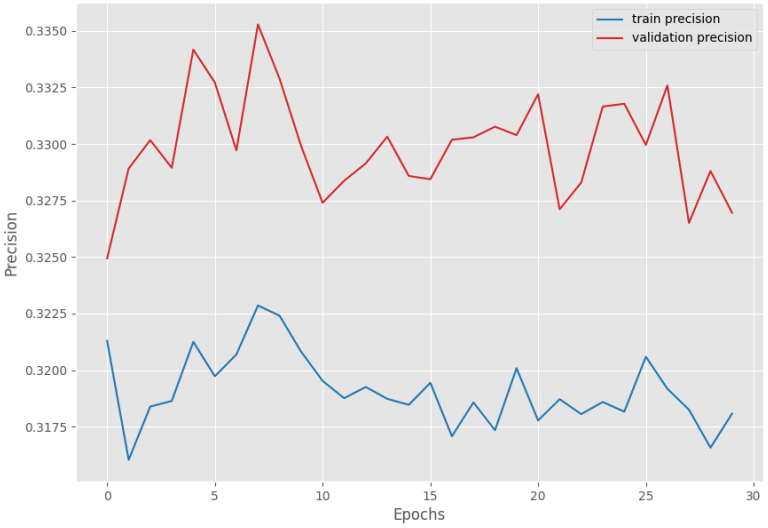


Figure 4.10: Precision using Mask2former for car class

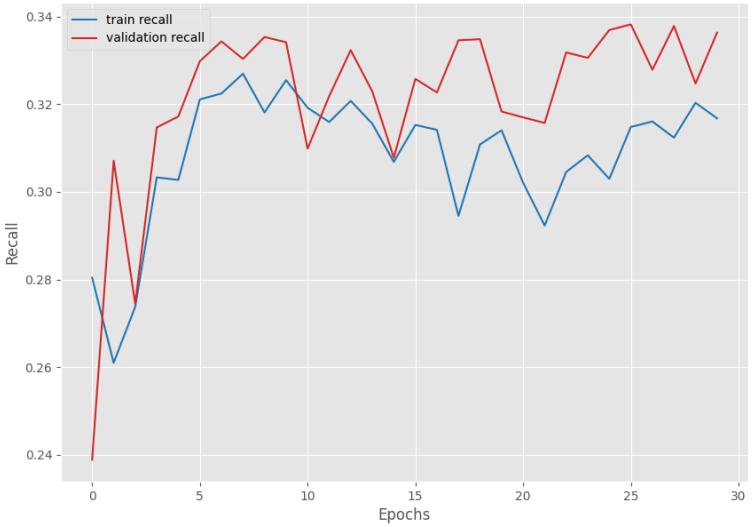


Figure 4.11: Recall using Mask2former for pedestrain class

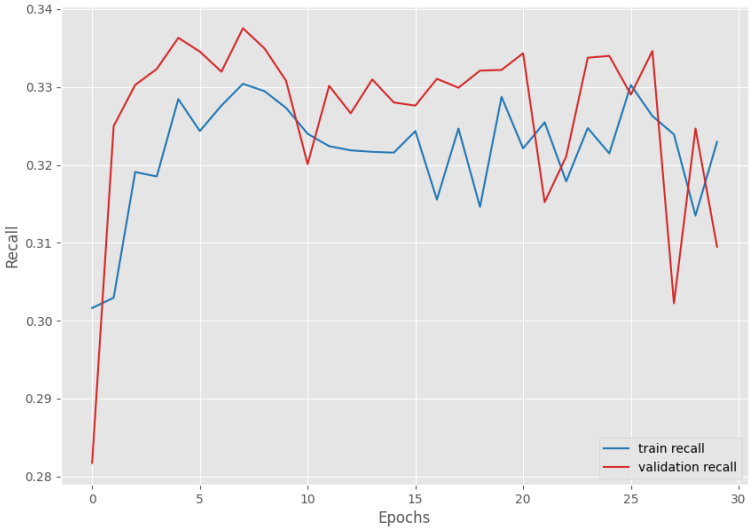


Figure 4.12: Recall using Mask2former for car class

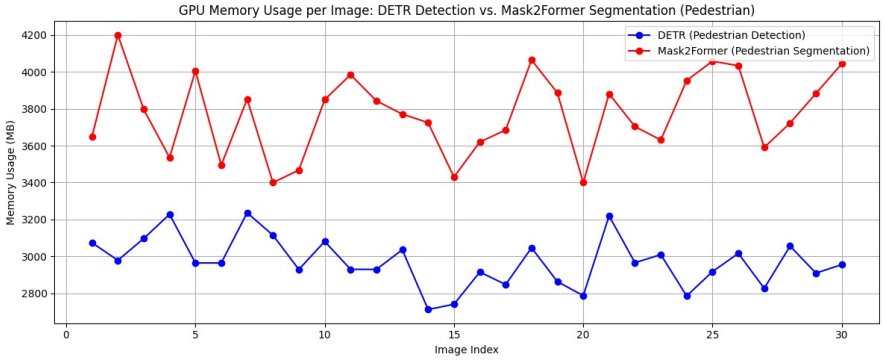


Figure 4.13: Memory usage for pedestrain class for both DETR and Mask2former

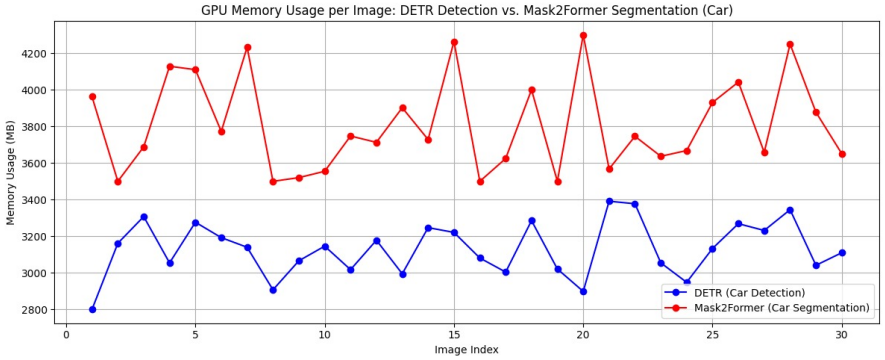


Figure 4.14: Memory usage for car class for both DETR and Mask2former

Figure 4.15: Result

Figure 4.16: Result

Figure 4.17: Result

CHAPTER 5

SUMMARY AND CONCLUSION

In this project, we applied transformer-based models DETR for object detection and Mask2Former for semantic segmentation on the CamVid dataset, focusing on the pedestrian and car classes. The system allows users to dynamically choose which class to detect and which to segment. We trained and evaluated the models using training loss, precision, and recall. The results showed better performance for the car class, likely due to its clearer appearance in the dataset. Speed plays a key role in this project because real time systems, like those used in self driving vehicles, must process visual information quickly to respond to traffic conditions. By allowing flexible selection of detection or segmentation per class, our approach reduces unnecessary computations and improves processing efficiency. This balance between accuracy and speed makes the system more practical for real-world deployment.

REFERENCES

[1] Carion, Nicolas, Francois Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. *End-to-End Object Detection with Transformers.* European Conference on Computer Vision (ECCV), 2020.

[2] Cheng, Bowen, Alexander Schwing, and Alexander Kirillov. *Masked-attention Mask Transformer for Universal Image Segmentation.* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1290–1299, 2022.

[3] Brostow, Gabriel J., Julien Fauqueur, and Roberto Cipolla. *Semantic Object Classes in Video: A High-Definition Ground Truth Database.* Pattern Recognition Letters, vol. 30, no. 2, pp. 88–97, 2009.

[4] Dosovitskiy, Alexey, et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.* International Conference on Learning Representations (ICLR), 2021.

[5] Xie, Enze, et al. *SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers.* Advances in Neural Information Processing Systems (NeurIPS), 2021.

[6] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2015.

[7] Lin, Tsung-Yi, et al. *Microsoft COCO: Common Objects in Context.* European Conference on Computer Vision (ECCV), 2014.