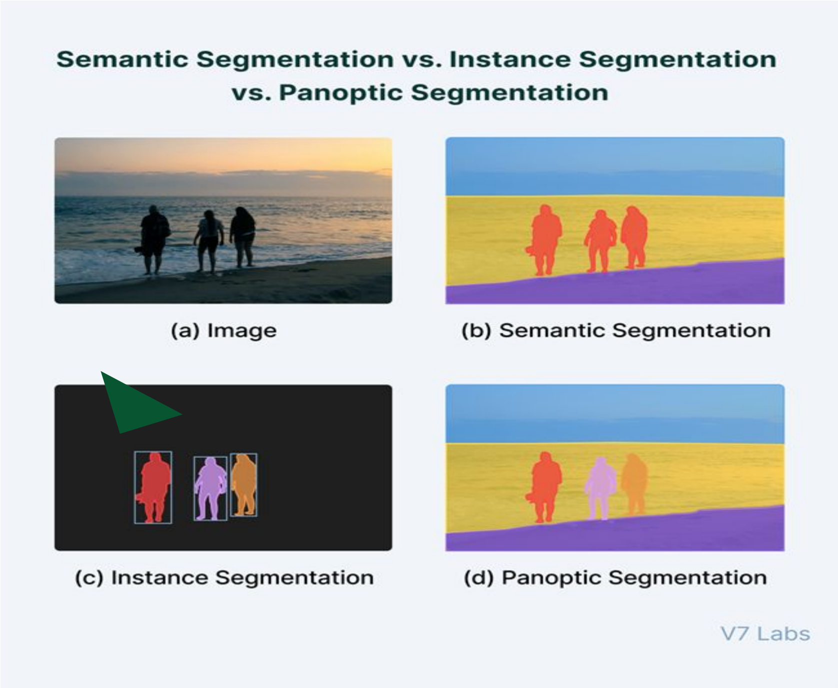
Panoptic Segmentation

Research Topic 12

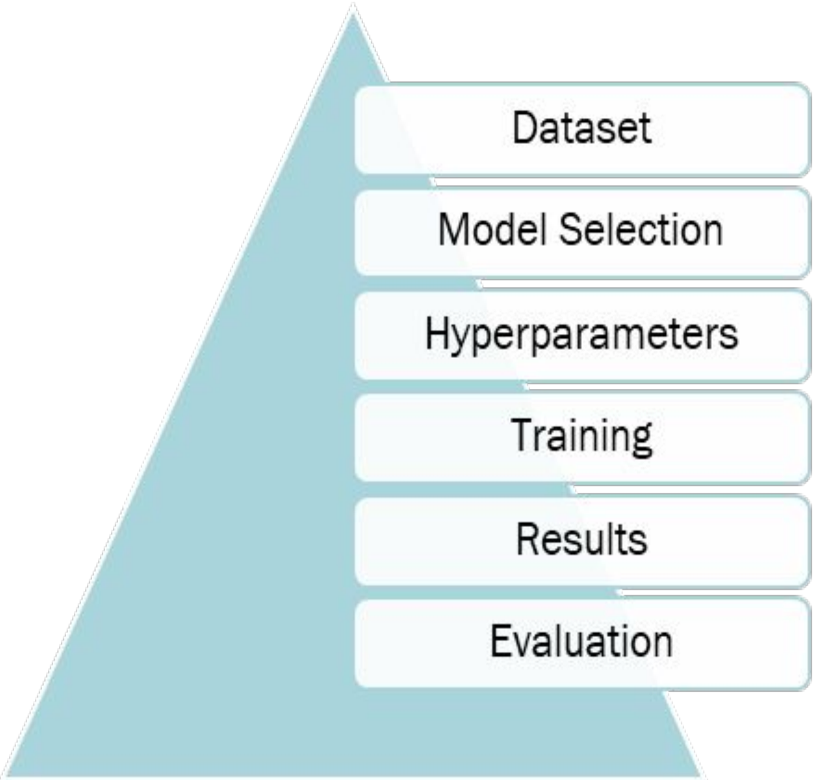
Natalia
Shubhom
Darpan

Introduction

- Panoptic Segmentation: A cutting-edge computer vision technique
- Key Emphasis: The fusion of Semantic and Instance Segmentation
- Result: The ability to perceive both what´s in an image and the quantity of each entity.
- Thing classes: contains both semantic and instance ID. Stuff classes: only semantic ID.



Methodology

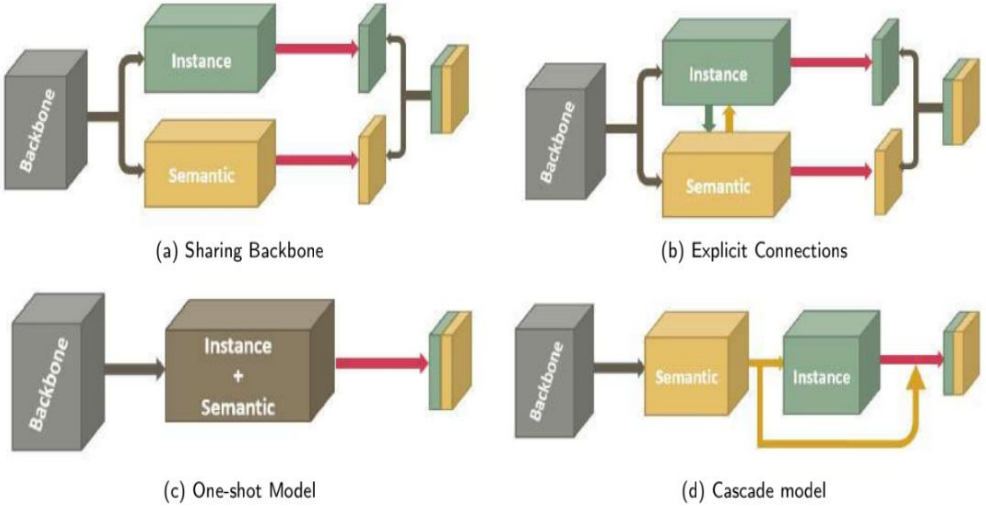


Datasets

1. **Cityscape Dataset**
    1. **Urban Insights:** Curated for urban scenes, it offers rich insights into city environments.
    2. **Diverse Scenarios:** Covers diverse weather conditions, traffic, and urban landscapes.
    3. **Semantic Segmentation:** Provides pixel-level semantic annotations.
    4. **Instance Segmentation:** Includes instance-level annotations for objects like cars and pedestrians.
    5. **Benchmarking:** Widely used for benchmarking and autonomous driving purpose.
2. **COCO Dataset**
    1. **Broad Scope:** Encompasses a wide variety of objects and scenes in everyday contexts.
    2. **Annotations:** Offers pixel-level annotations for both semantic and instance segmentation.
    3. **Object Detection:** Includes object detection annotations for 80 object categories.
    4. **Benchmark Leader:** Often used as a benchmark for object detection, segmentation, and panoptic segmentation.
    5. **Challenges:** Stimulates the development of novel algorithms and models.

Model selection

- Four Methodologies to perform Panoptic Segmentation
  - Sharing Backbone
  - Explicit Connection
  - One Shot model
  - Cascade model
- One shot method uses one single network for panoptic making it more efficient in terms of model size and run time.
- Some of the popular One-shot methods are DETR, Maskformer, Mask2former Panoptic FCN etc.
- One common thing noted in most of these model is the use of transformer based model.



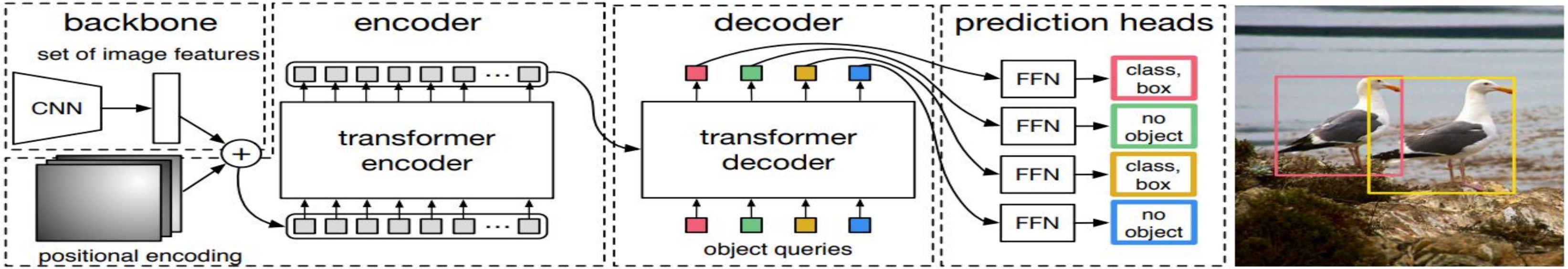
(a) Sharing Backbone (b) Explicit Connections (c) One-shot Model (d) Cascade model

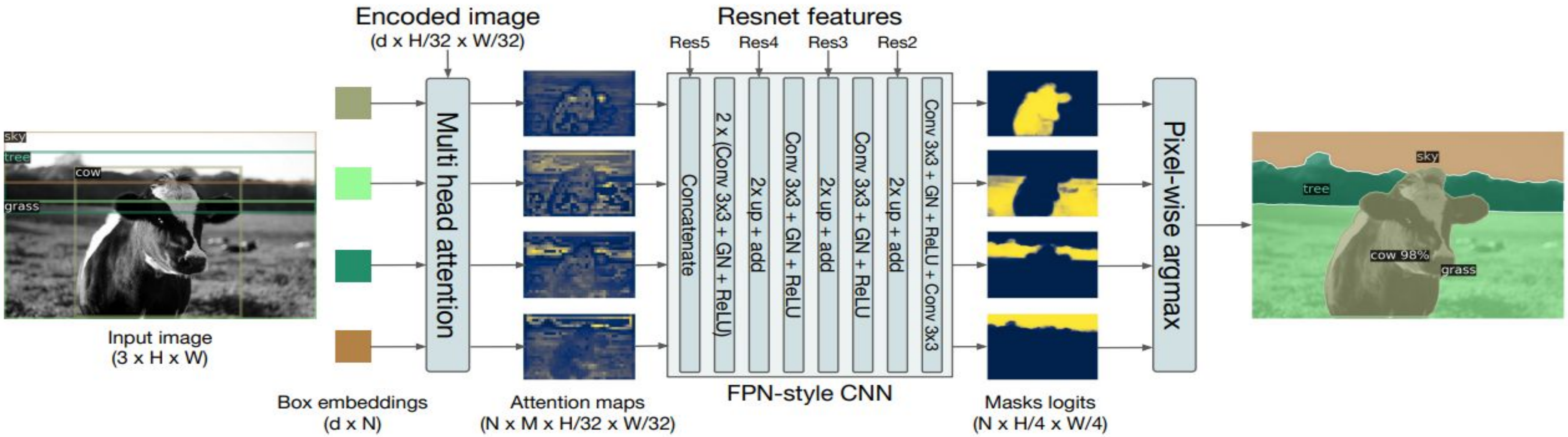
Model selection: Transformer based models

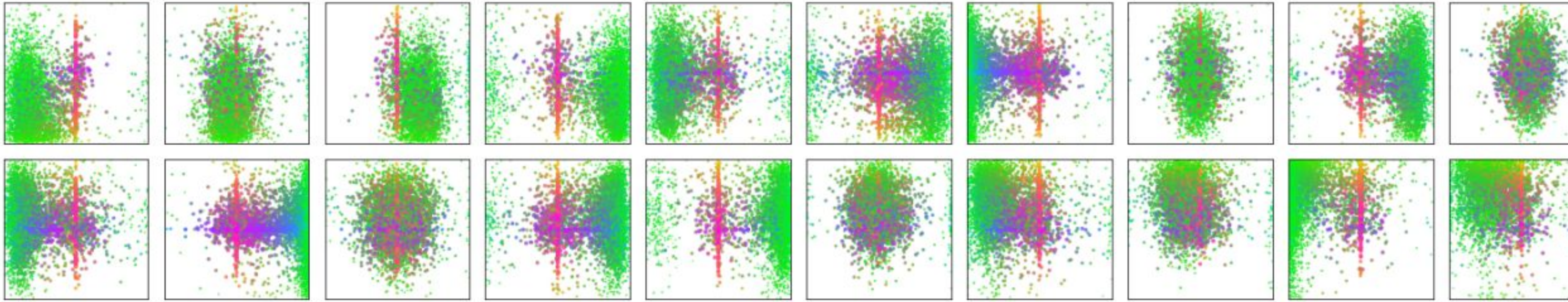
- Some of the transformer based model that were chosen initially were Maskformer, Mask2former and DETR.
- All of these models are mask classification models used to set binary mask belonging to various classes.
- Previous approaches were pixel-wise classification.
- The mask based approach helps in unifying the model with common loss and training procedure for both semantic and instance part of segmentation
- DETR was chosen due to its simple and elegant approach towards panoptic segmentation. The resource availability for the model was a crucial criteria.

DETR: DEtection TRansformer

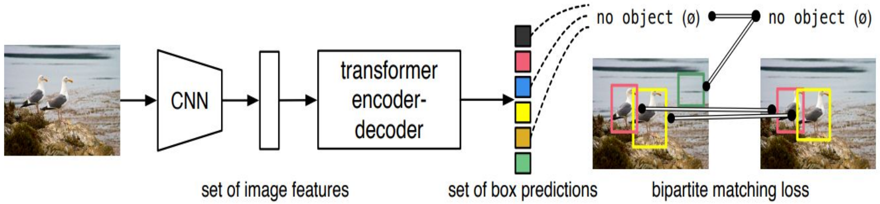
- DETR was among the first transformer based computer vision model introduced by Facebook AI.
- The model consists of four key elements
  - Backbone: used for feature extraction
  - Encoder-Decoder transformer architecture: To generate output embeddings
  - Feed forward network: for generating regression output
  - Mask head: for converting feature maps to panoptic output




Encoded image
(d x H/32 x W/32)
Resnet features
Res5
Res4
Res3
Res2
sky
tree
cow
grass
Input image
(3 x H x W)
Multi head attention
Concatenate
2 x (Conv 3x3 + GN + ReLU)
2x up + add
Conv 3x3 + GN + ReLU
2x up + add
Conv 3x3 + GN + ReLU
2x up + add
Conv 3x3 + GN + ReLU + Conv 3x3
Pixel-wise argmax
sky
tree
cow 98%
grass
Box embeddings
(d x N)
Attention maps
(N x M x H/32 x W/32)
FPN-style CNN
Masks logits
(N x H/4 x W/4)



Key elements of DETR Model



- Upon the advantages from the transformer model like positional encoding , the detr model has other important elements.
- DETR model does not require use of non maximum suppression since it uses learned object queries and Bipartite Matching Loss.
- Unlike conventional object detectors that rely on predefined anchor boxes, DETR uses learned object queries. Number of object queries define the number of object detections in a single pass
- DETR uses a cross-attention mechanism to associate objects in the image with object queries.
- DETR employs a specialized loss function called the "Hungarian loss" or "Bipartite Matching Loss" to associate predicted detections with ground truth objects, ensuring that the model learns to correctly assign objects to queries.

$$\hat{\sigma} = \underset{\sigma \in \mathfrak{S}_N}{\arg\min} \sum_{i}^{N} \mathcal{L}_{\mathrm{match}}(y_i, \hat{y}_{\sigma(i)}),$$

$$\mathcal{L}_{\mathrm{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\mathrm{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right]$$

Data Augmentation

- Two categories:
    - Geometric Augmentation   Eg. Flipping, Rotations, Zooming etc
    - Color space augmentation   Eg Brightness, Saturation, contrast etc
- We decide to go with Color space augmentation as creation of new labels was not needed. Existing labels were mapped to both previous and the augmented image.
- Four types: Brightness, Saturation, Contrast and pixel quality were varied for the augmentation process.
- Good probabilistic policy was chosen in deciding the types of augmentation to be applied and the amount of the chosen augmentation on the image.

Model

DETR-Panoptic

1. •**End-to-End Learning:** Simultaneously handles object detection and panoptic segmentation tasks within a single model.
2. •**Attention Mechanisms:** Effectively captures global context information.
3. •**Object Detection:** Demonstrates remarkable object detection performance.

Mask2Former

1. •**Transformer Architecture:** Utilizes a transformer-based architecture for strong contextual understanding.
2. •**Semantic and Instance Segmentation:** Seamlessly fuses semantic and instance segmentation information.
3. •**Fine-Grained Segmentation:** Excels in providing detailed segmentations for complex scenes.

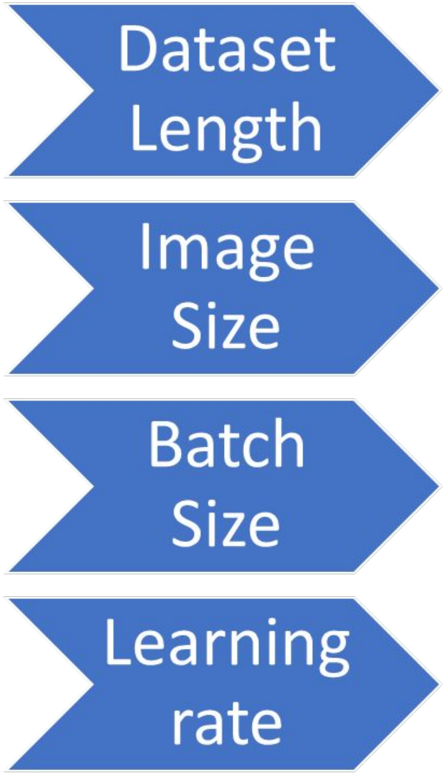
MaskFormer

1. •**Hybrid Model:** Combines the strengths of transformers and convolutional neural networks (CNNs).
2. •**Object Detection:** Capable of efficient object detection.
3. •**Segmentation Quality:** Provides high-quality panoptic segmentations.

EfficientPS

1. •**Efficiency:** Focuses on lightweight and efficient panoptic segmentation.
2. •**Real-Time Applications:** Suitable for real-time or resource-constrained applications.
3. •**Balanced Performance:** Maintains a balance between accuracy and computational requirements.

Hyperparameters



Training

- **Dataset Length:** Total approx. 3000 images, we have trained with 1000,1200 and 2975.
- **Exploration of Learning Rate:** For both the primary learning rate and the supporting learning rate, we looked at learning rates like 10-4, 10-3, and 10-5. We were able to find the combination through rigorous testing that reached the ideal mix between training loss convergence and segmentation quality.
- **Batch Size Choice:** We tested with batch sizes of 4, 6, and 8 due to system memory limitations. We found that batch sizes above this range would result in memory restrictions, which would affect the training process.
- **Image Size:** One essential hyperparameter was image size. We looked examined 400, 500, 600, and 700-input sizes. Memory usage started to become an issue above an input size of 800, making the system unusable for training. We had to downsample to size of 700.
- **Maximum Image Size:** We looked at several maximum picture sizes, similar to image size. To balance model performance and memory efficiency, this parameter was changed.
- **Gradient clipping:** With values of 0.1 and 1.0, gradient clipping was tested. We chose 0.1 instead of 1.0 since 1.0 produced unfavorable results while 0.1 increased training stability.
- **Optional Activation Function:** We tested the classification head's ReLU activation mechanisms. Surprisingly, this decision had poor performance, which led us to go back to the original activation functions in order to get better outcomes.

Training

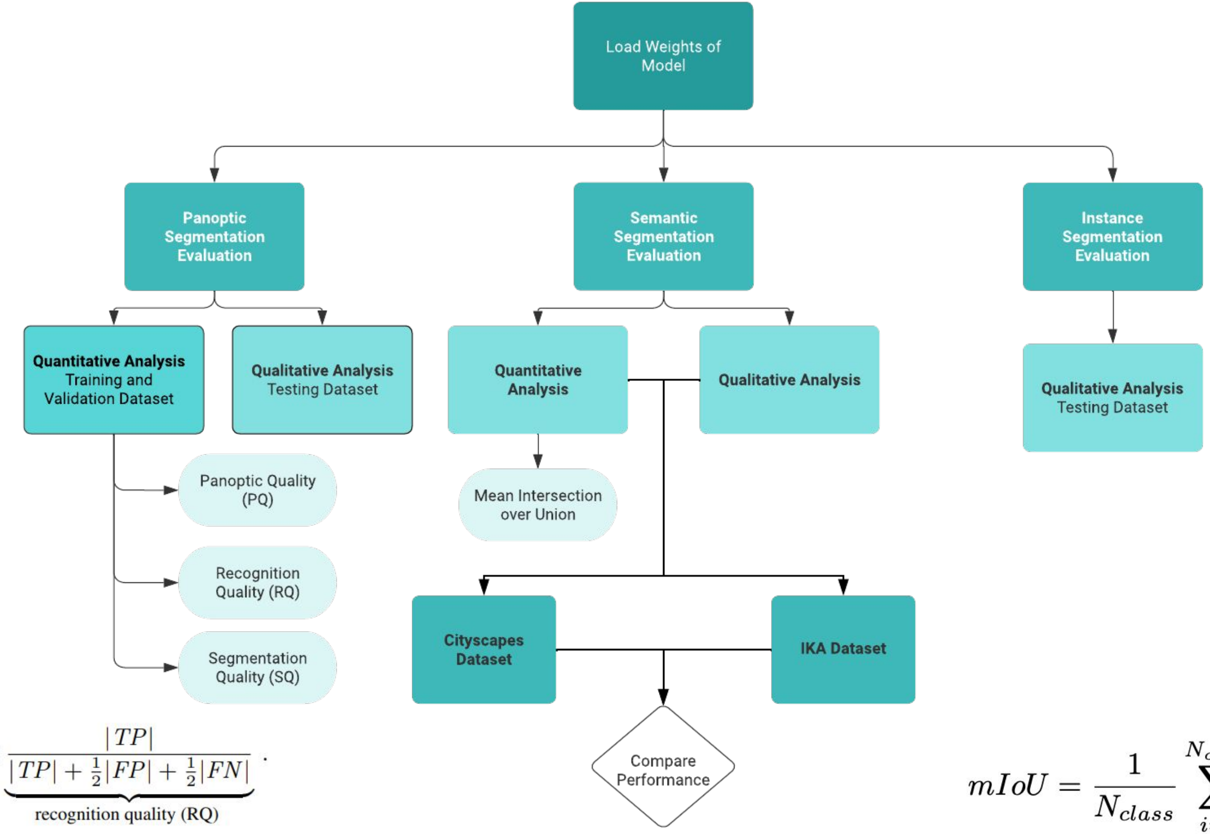
Local System Experiments:

- Local system configuration: Nvidia RTX 3070TI GPU with 8GB of graphics card memory and 16GB of RAM. This configuration was inferior to that of the server.
- Given the limitations of our local system compared to the server, we decreased our hyperparameters to align with its capabilities. These adaptations allowed us to find a configuration that worked best for our system.

Server Experiments:

- One significant challenge we encountered during server-based training was the fluctuating GPU memory. The varying GPU memory caused frequent kernel crashes as the initially assigned memory for the model reduced throughout the day. Consequently, we had to adjust the batch size for every run to match the available GPU memory, resulting in longer-than-expected training times.

Evaluation Procedure Overview



$$\mathrm{PQ} = \underbrace{\frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}}.$$

$$mIoU = \frac{1}{N_{class}} \sum_{i=1}^{N_{class}} \frac{TP(i)}{TP(i) + FP(i) + FN(i)}$$

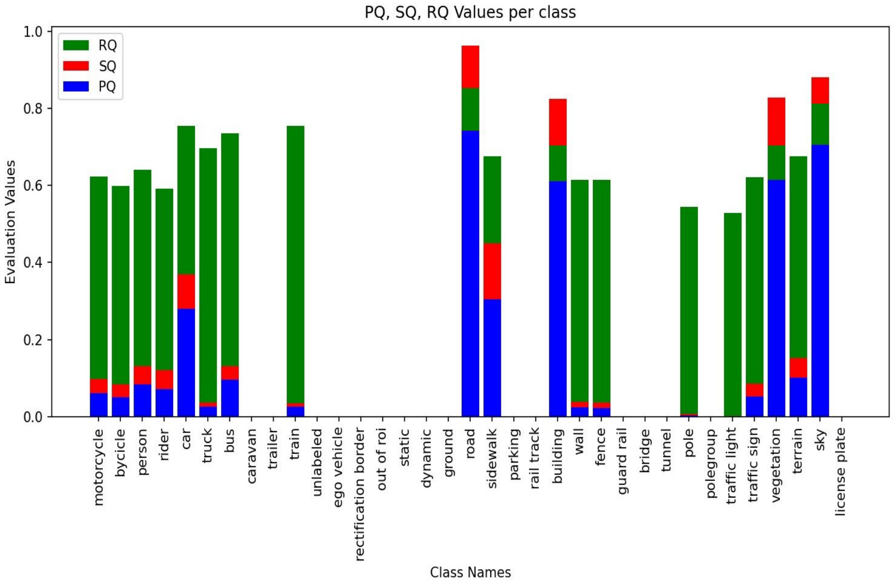
Models Evaluated

Parameter	Value
Train Dataset Size	1200
Test Dataset Size	50
Validation Dataset Size	50
Image Size	400
Longest Edge	400
Learning Rate	1×10^{-4}
Learning Rate Backbone	1×10^{-4}
Number of Epochs	50 (v125 w/ data aug) 100 (v119)
Train Batch Size	4
Test Batch Size	2
Validation Batch Size	1
Hidden Units	[128, 64]
Dropout Rate	0.2
Optimizer	Adam
Weight Decay	1×10^{-4}
Gradient Clip Value	0.1

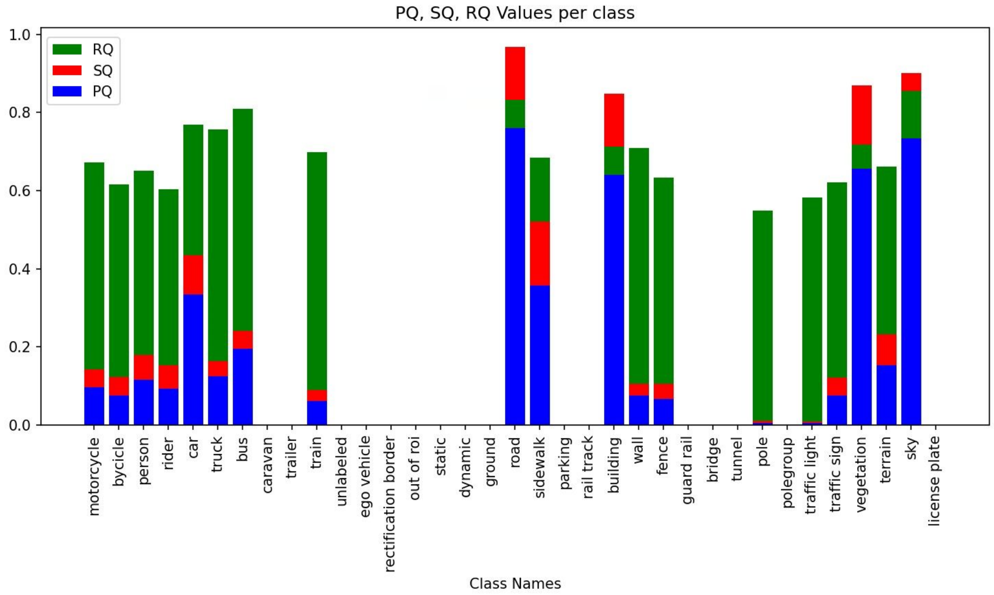
Model	Metrics			
	PQ	SQ	RQ	MIoU
v125 with Data Augmentation	5.48	36.38	15.06	16.22
v119	6.66	37.51	17.76	18.63

Panoptic Segmentation Results (1/2) - Visualisation of class-wise performance

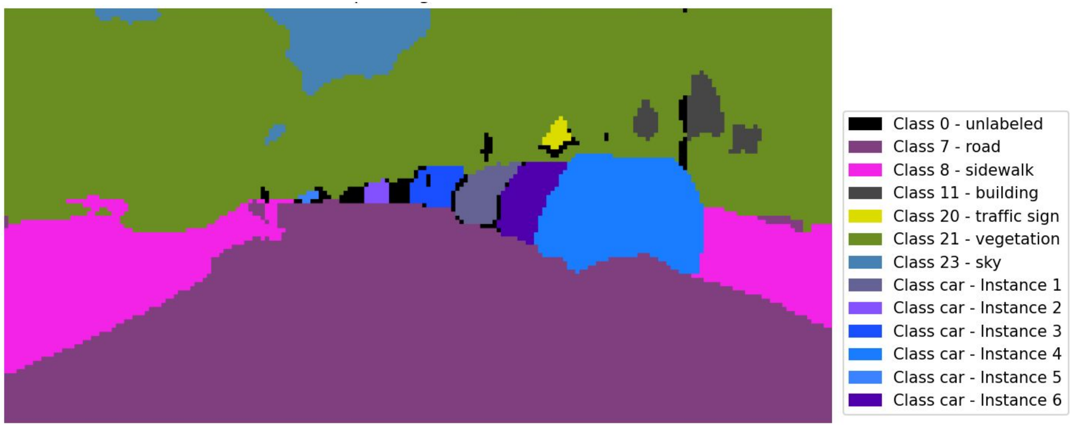
Model 125 with data augmentation

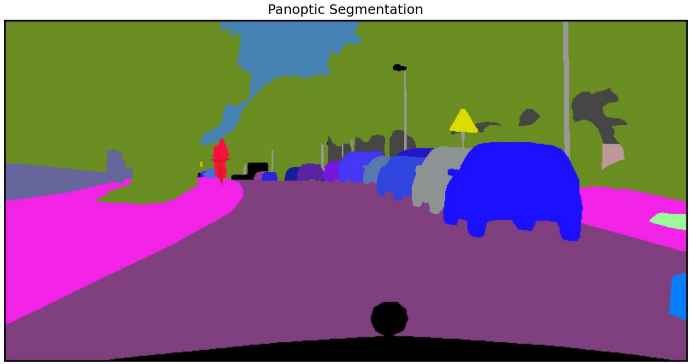


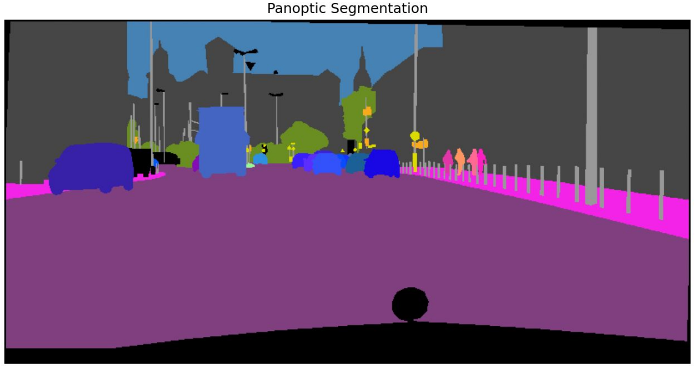
Model 119



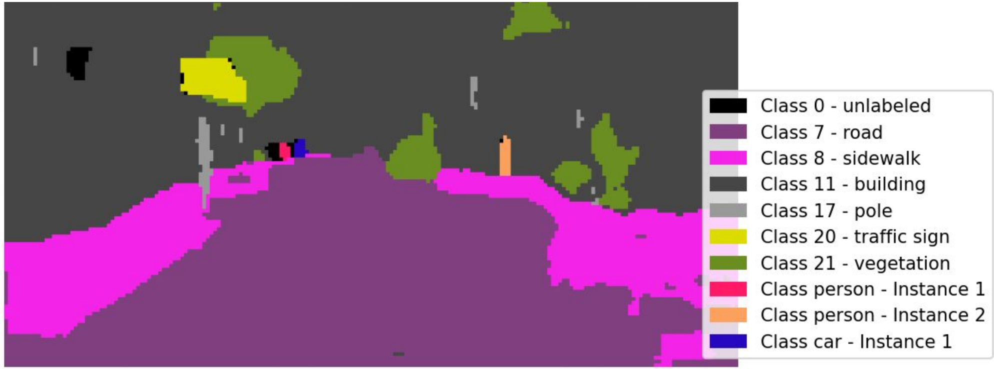
Visual Analysis v119 (1/2) - Panoptic Segmentation



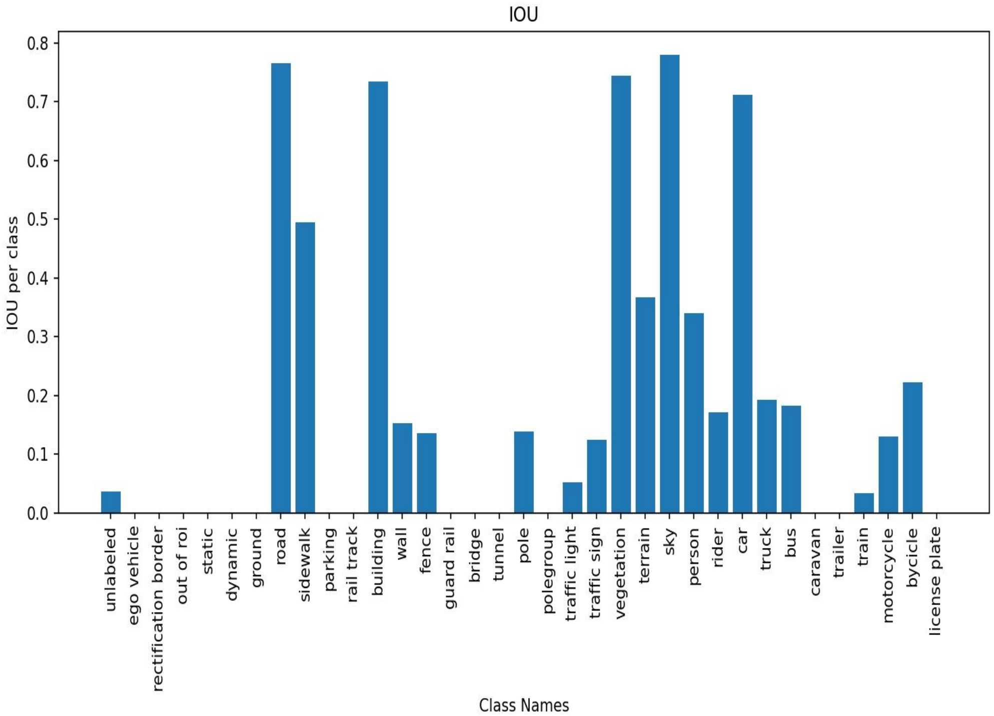


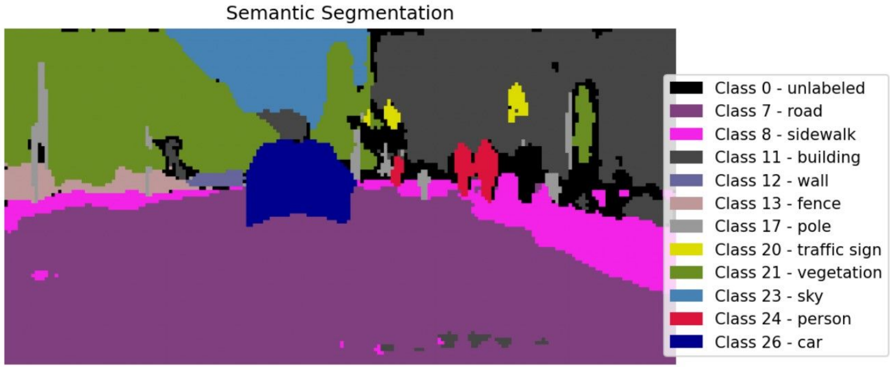


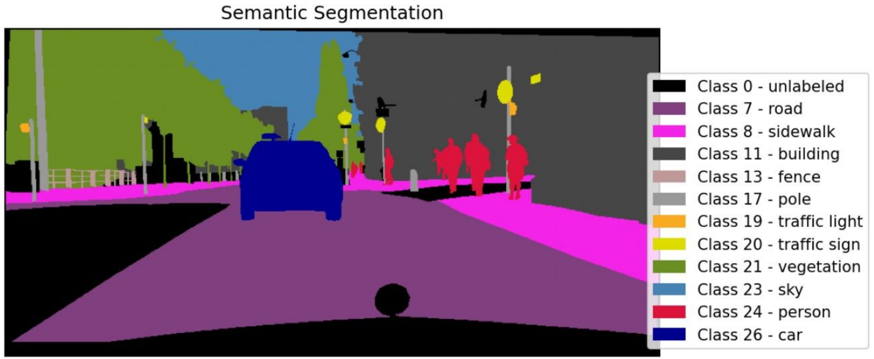
Visual Analysis v119 (2/2) - Limitations of Model



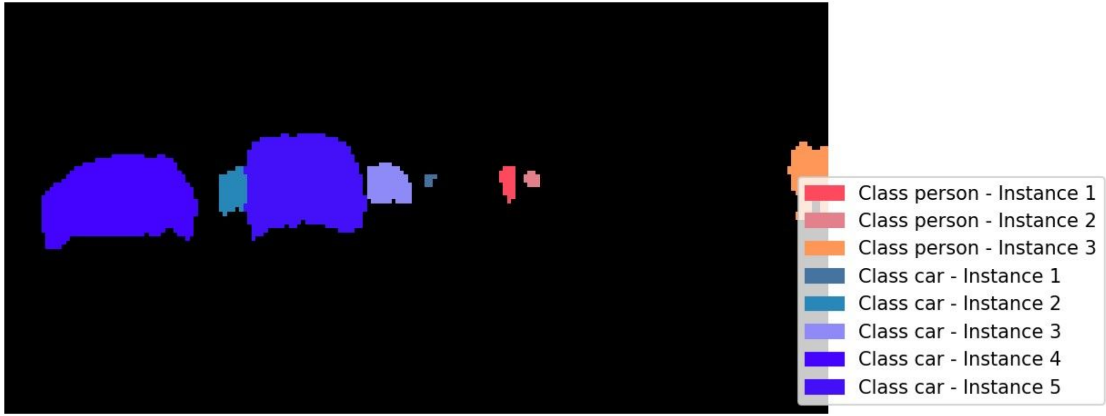
Semantic Segmentation Performance

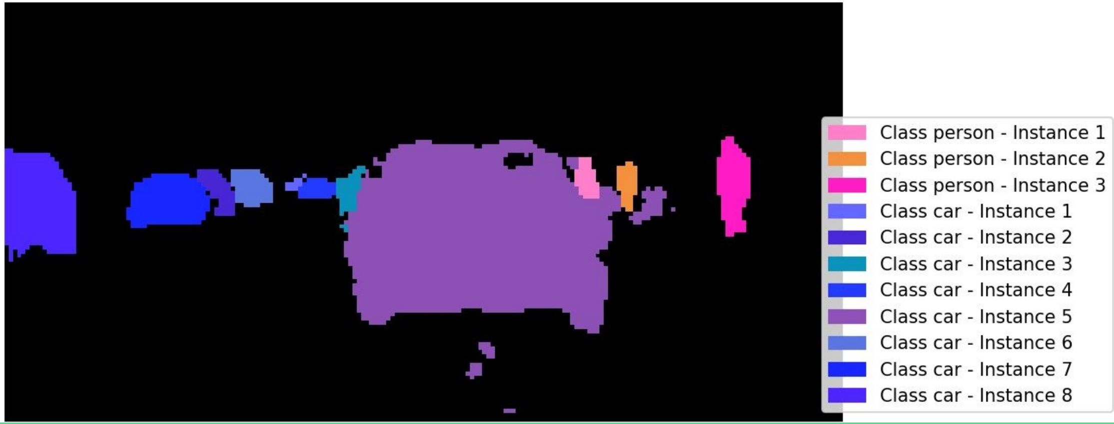




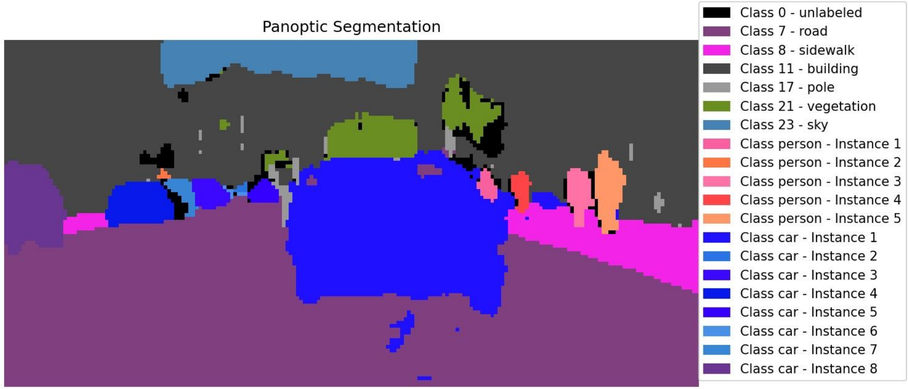


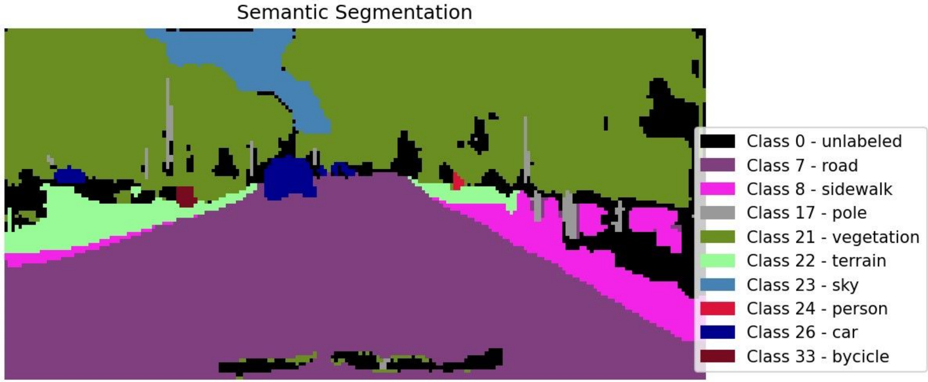
Instace Segmentation Performance

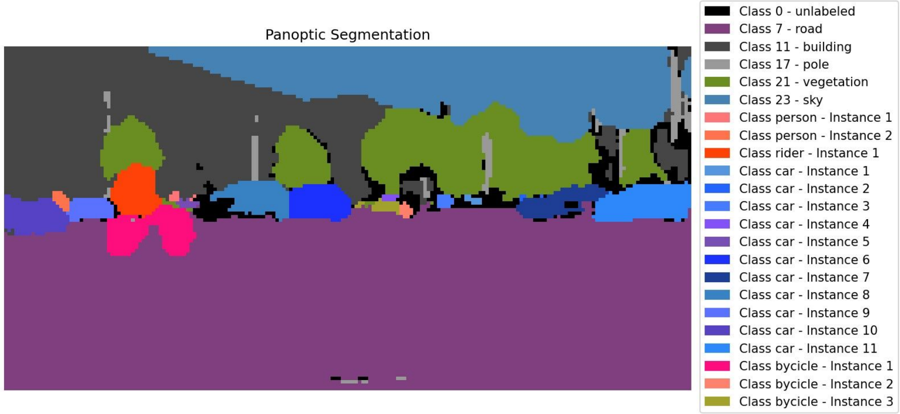


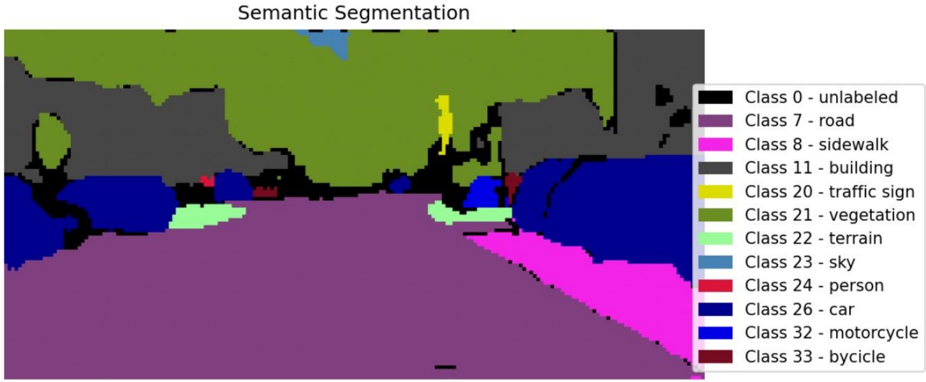


Generalisation Capabilities - Extrapolation to Test Dataset

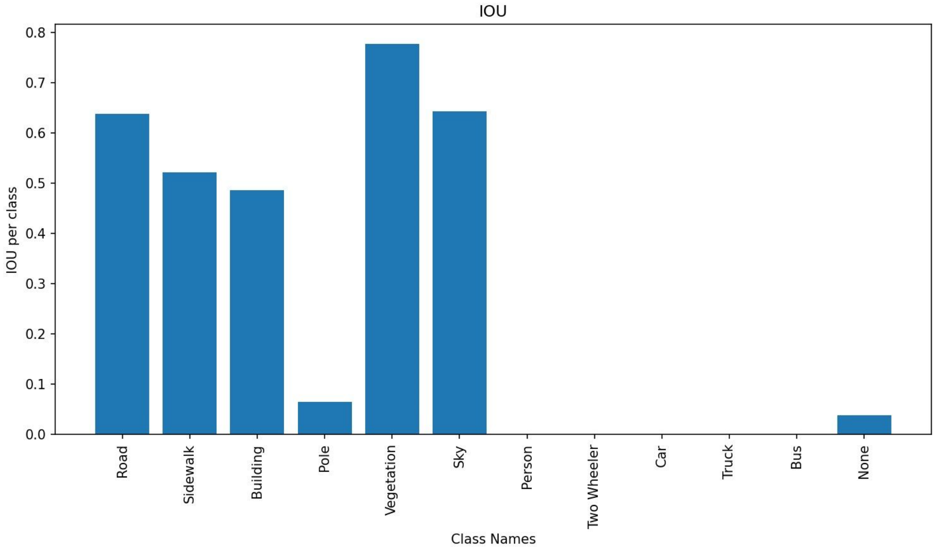


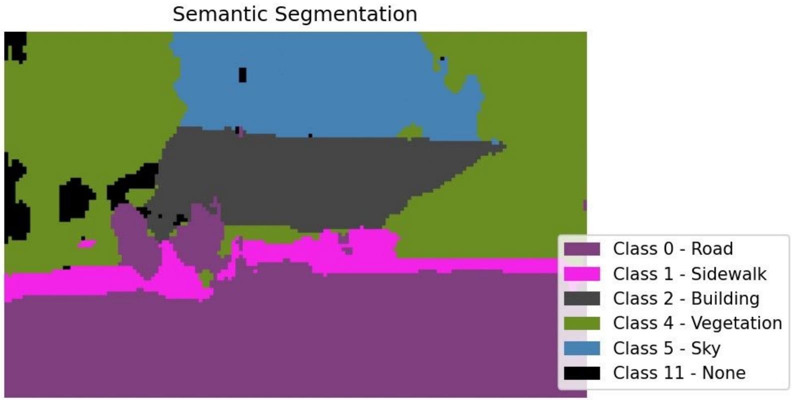


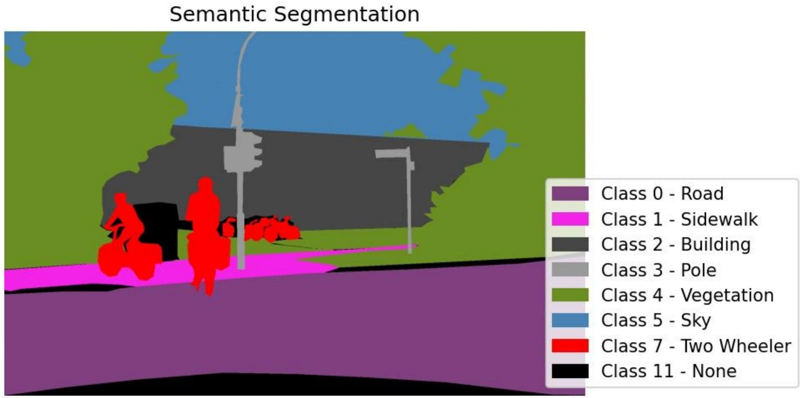




Generalisation Capabilities - IKA Dataset



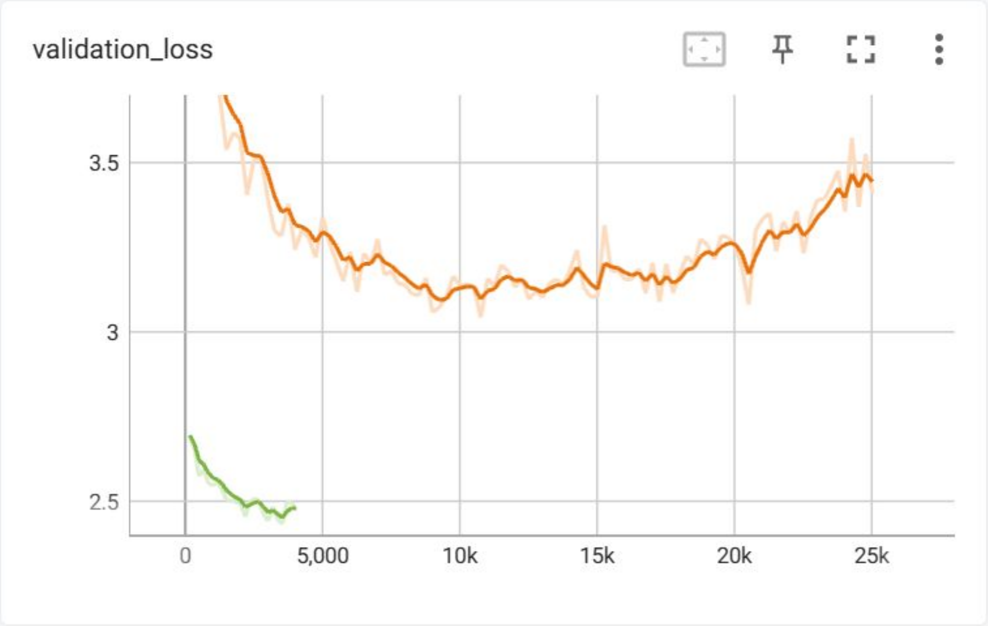


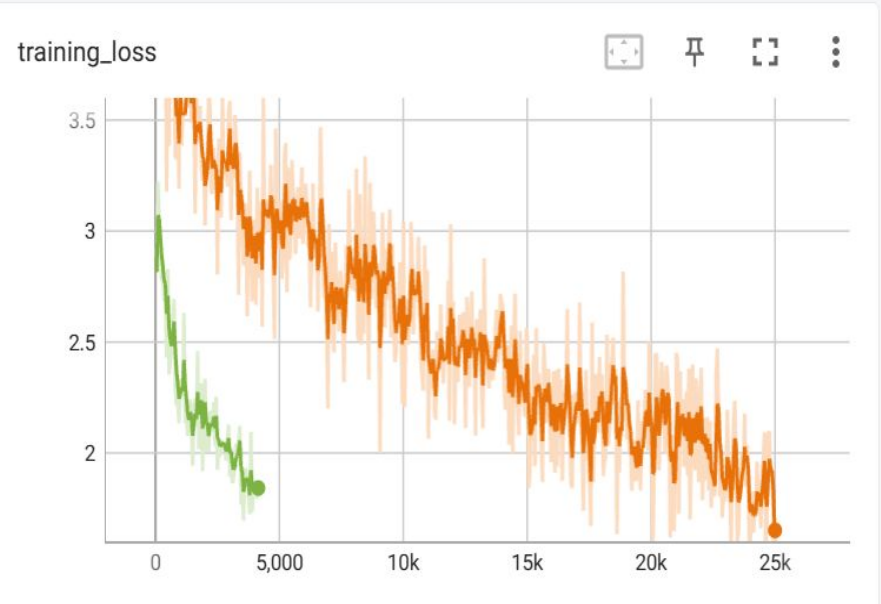


Dataset	IKA	Cityscapes
MIoU Value	26.56	18.63

Results-1(Dataset Comparison)

Validation Loss



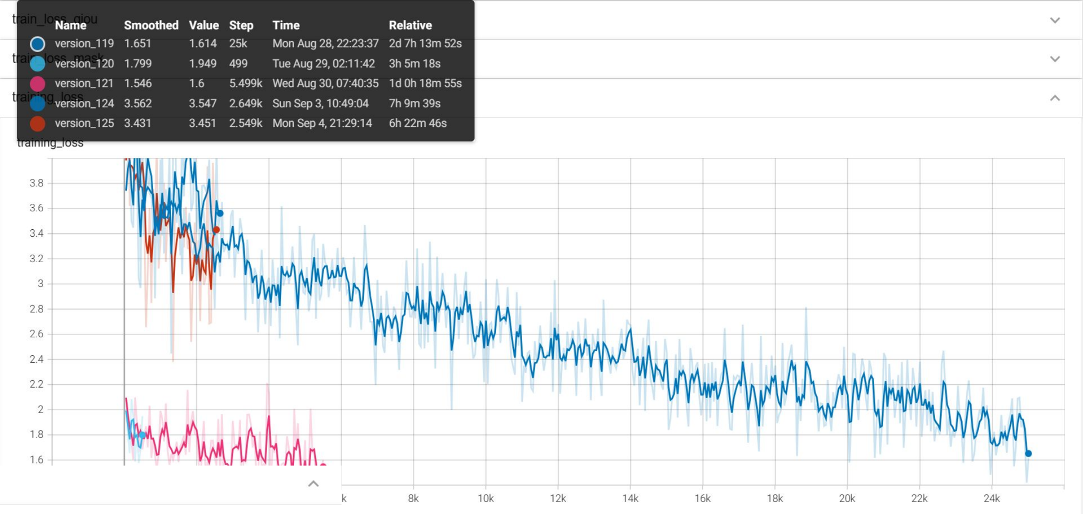


Training Loss

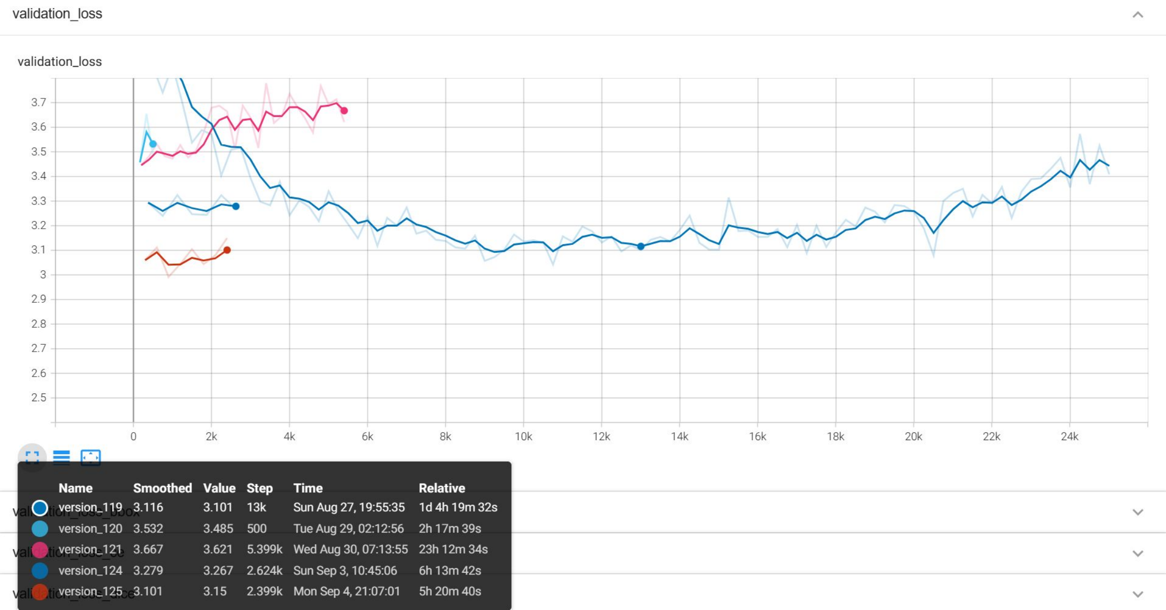
Model 97_V2 (2975 images)
Model 119(1200 images)

Results-2 (Augmentation)

119-Without Augmentation
125-With Augmentation



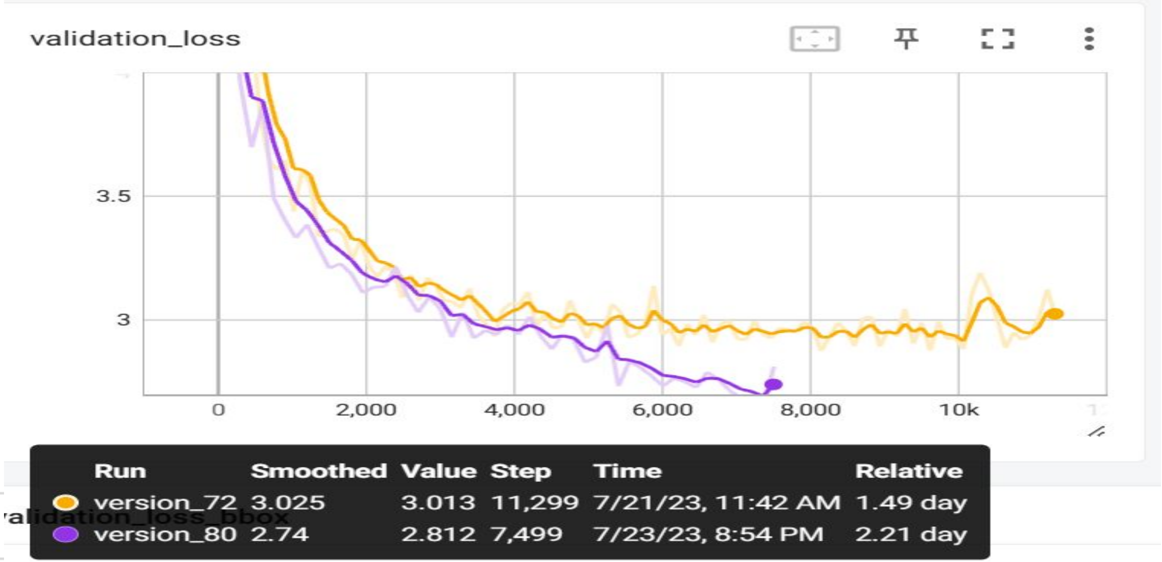
Validation Loss

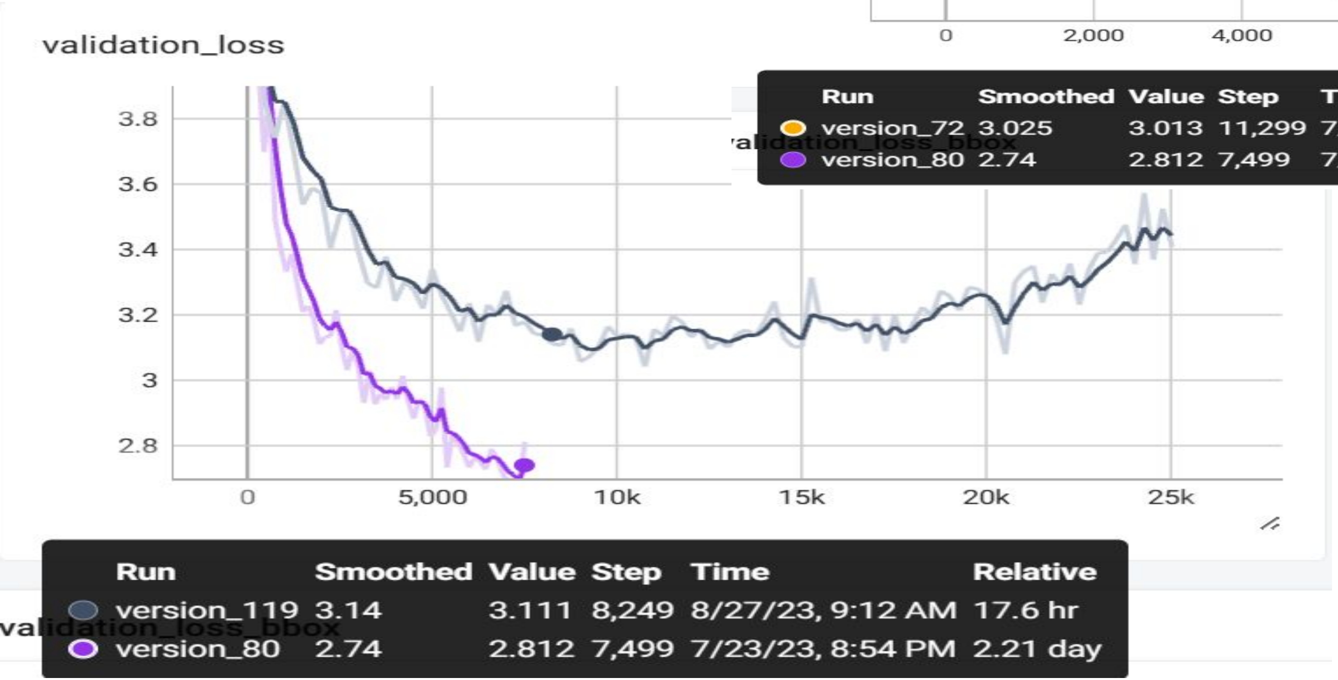


Training Loss

Results-3

80- Older Collate Function
119- New Collate Function

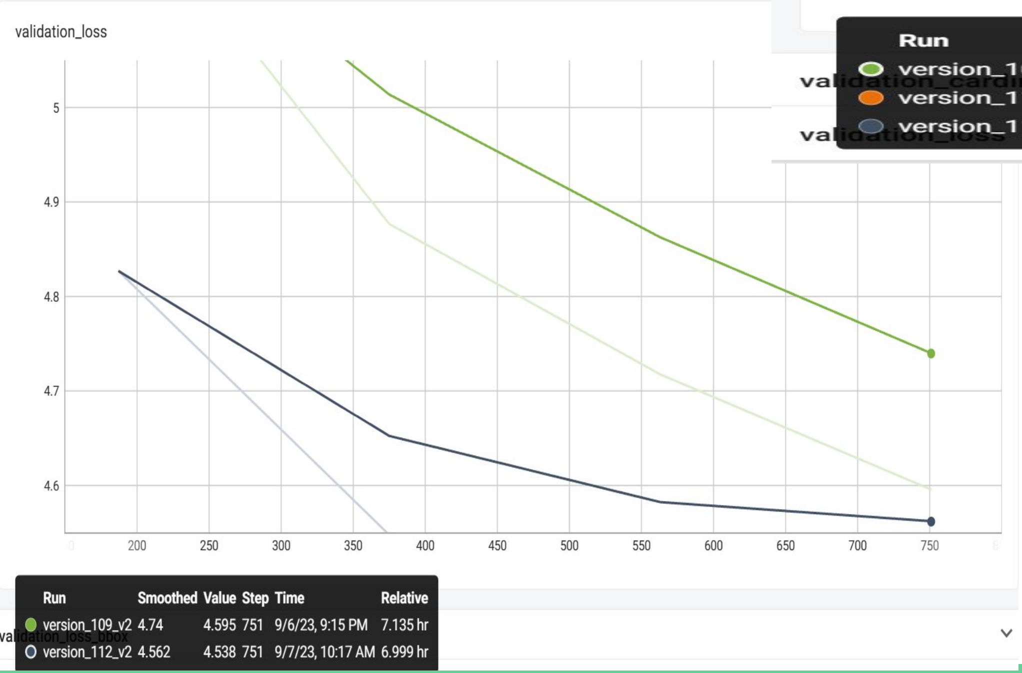




80- Without ReLU
72- With ReLU

Results-4

Transformer freezing



Limitations

- **Data Set Limitations:** We are not utilizing full dataset ,because of system limitations. This leads to higher training time.
- **Annotation Quality:** Due to inaccuracies or ambiguities in labeling, which can affect model training.
- **Instance Boundary Challenges:** Difficulty of accurately delineating instance boundaries, especially for smaller or complex objects, and the resulting impact on instance segmentation quality.
- **Class Reduction in IKA Dataset: Domain Shift:**Reduction in the number of classes in the IKA dataset simplifies training but also results in a loss of detailed semantic information compared to the original Cityscapes dataset.

Limitations-2

- **Hyperparameter Sensitivity:** The fine-tuning process can be time-consuming and demanding.
- **Lack of Universality:** Optimal hyperparameters may vary for different datasets and applications.
- **Class Imbalance:** Classes are not equally distributed. Focal loss is being used but it could have been used with better class distribution.
- **Visual vs. Quantitative Trade-off:** Sometimes there's a trade-off between achieving lower quantitative loss metrics and obtaining visually pleasing segmentations.
- **System Limitations:**Due to varying GPU memory hyperparameters like batch size and image size needs to be decreased to meet the system requirements.

```
cuda_mem()

Total memory: 23.70 GB
Free memory: 0.22 GB
Used memory: 23.47 GB
```

```
cuda_mem()

Total memory: 23.70 GB
Free memory: 8.58 GB
Used memory: 15.12 GB
```

Future Scope – Improvement and Different approach

- **Utilizing Full Dataset:**Leveraging the entire dataset to enhance model robustness and generalization.
- **Data Augmentation:** Geometric augmentation techniques can be used to diversify the training data and improve model adaptability.
- **Increasing Batch Size:**Discussing the advantages of larger batch sizes in terms of training speed and convergence stability.
- **Increasing Image Size:** Analyzing the impact of larger input image sizes on model performance and its suitability for different scenarios.
- **Changing Model Architecture:** Evaluating the potential of alternative model sections to address panoptic segmentation challenges.
- **Freezing Different Layers:** Exploring the effects of freezing specific layers in the model to optimize training dynamics and model adaptability.


Questions?