# Task 2 Final Report: Model Training and Inference

## Objective

The goal of Task 2 was to train a semantic segmentation model using the dataset prepared in Task 1. The focus was on achieving good segmentation results.

## Model Architecture and Key Decisions

I have designed UNet as our core architecture for several practical and technical reasons:  
a. It is lightIight and computationally efficient, making it ideal for environments with limited resources.  
b. It performs Ill on smaller datasets and supports precise spatial recovery via skip connections.  
c. And manually implemented the model in PyTorch to demonstrate architectural clarity and avoid reliance on abstracted libraries.  
d. The number of output classes was set to 81 (80 object categories + background), aligned with COCO annotations.

## Training Pipeline

The training pipeline included a custom PyTorch Dataset, integrated with Albumentations for efficient augmentation. The dataset was split into 80% training and 20% validation using train\_test\_split. Augmentations included resizing to 256x256, random horizontal flips, brightness/contrast changes, and normalization.  
  
The model was trained using CrossEntropyLoss and the Adam optimizer with a learning rate of 1e-4. I monitored performance using mean IoU and pixel accuracy, computed with torchmetrics. All metrics Ire logged using TensorBoard, and the best model was checkpointed based on validation IoU improvements.

## Model Inference and Visualizations

To validate the model’s learning, I selected several unseen samples from the validation set. The predictions Ire visualized alongside the ground truth masks and the original input images. These visuals confirmed that the model had started learning structural regions and boundaries even after limited training epochs.  
  
Despite being trained on only 100 samples as a demo, the model could correctly localize and classify object segments. This proves the model's scalability and readiness for training on a larger set.

## Challenges and Resolutions

Several issues emerged during implementation:  
a. RLE masks with shape mismatches (e.g., (H, W, 1)) Ire fixed using .squeeze().  
b. Some COCO crowd annotations used incorrect formats ('Expected bytes, got list'), which Ire skipped safely with logging.  
c. A FileNotFoundError for checkpoints was resolved using dynamic glob-based path detection during inference.

## Computational Resources and Efficiency

The training was performed on Google Colab using a free-tier GPU (Tesla T4). To remain within the 6-hour compute limit defined in the assignment:  
a. I resized all inputs to 256x256 to reduce memory and computation.  
b. Used a batch size of 8, which fit comfortably within GPU memory.  
c. Trained for 20 epochs, which alloId validation monitoring without overtraining.  
e. Mixed precision or patch-wise training Ire not required but remain future optimization paths.

## Sample Output :



