# Deep Learning for Semantic Segmentation and Object Detection

Djahid ABDELMOUMENE

March 17, 2025

# Part I: Semantic Segmentation

## 1 Introduction to Semantic Segmentation

Semantic segmentation assigns a class label to each pixel in an image. You will explore a deep learning model for semantic segmentation: U-Net. The goal is to accurately label each pixel, thereby partitioning the image into semantically meaningful regions. Common evaluation metrics include Intersection over Union (IoU) and Pixel Accuracy.

## 2 Datasets

For semantic segmentation, you may use a subset of the COCO dataset or any publicly available dataset that provides pixel-level annotations. Ensure the images are preprocessed appropriately (e.g., resized to 128x128 or 256x256) and normalized.

## 3 U-Net

### 3.1 Overview

U-Net is an encoder-decoder architecture originally designed for biomedical segmentation. Its skip connections help to preserve fine-grained spatial information lost during downsampling.

### 3.2 Architecture Details

- **Contracting Path:** Series of convolutional layers with pooling to capture context.
- **Expansive Path:** Upsampling layers combined with skip connections from the encoder, which help to localize and refine the segmentation map.

### 3.3 Implementation Considerations

- Use convolutional layers with ReLU activations.
- Apply batch normalization to stabilize training.
- Use a final 1x1 convolution to map to the desired number of classes, followed by a softmax (or sigmoid for binary segmentation).

## 4 Training Process and Evaluation

### 4.1 Training

- Preprocess and split the dataset into training, validation, and testing sets.
- Choose loss functions (e.g., Cross-Entropy and Dice Loss) and optimizers (e.g., Adam).
- Monitor training with loss curves and segmentation overlays.

### 4.2 Evaluation Metrics

- **Intersection over Union (IoU):** Measures the overlap between predicted and ground truth masks.
- **Pixel Accuracy:** The proportion of correctly classified pixels.

# Part II: Object Detection

## 5 Introduction to Object Detection

Object detection not only classifies objects within an image but also localizes them with bounding boxes. In this section, we explain a detailed, step-by-step approach to an object detection algorithm. We use Faster R-CNN as the primary example.

## 6 Overview of Faster R-CNN

Faster R-CNN is a two-stage object detection framework that integrates region proposal generation with object classification and bounding box regression. Its pipeline includes:

**Step 1: Feature Extraction:** A convolutional neural network (e.g., ResNet) extracts feature maps from the input image.

**Step 2: Region Proposal Network (RPN):** Uses sliding windows over the feature maps to propose candidate object regions (anchors).

**Step 3: RoI Pooling:** Regions of interest (RoIs) are pooled into a fixed-size feature map.

**Step 4: Classification and Regression:** Fully connected layers classify the object and refine the bounding box coordinates.

## 7 Detailed Step-by-Step Algorithm

### 7.1 Feature Extraction

- Input the image into a backbone network (e.g., ResNet-50) to extract high-level feature maps.

### 7.2 Region Proposal Network (RPN)

- Slide a small network over the feature maps to generate region proposals.
- For each sliding window location, generate multiple anchor boxes with various scales and aspect ratios.
- Classify anchors as foreground (object) or background and refine anchor coordinates.

### 7.3 RoI Pooling

- Crop and resize the proposed regions (RoIs) from the feature map to a fixed size using RoI pooling.

### 7.4 Object Classification and Bounding Box Regression

- Feed the pooled features into fully connected layers.
- Use softmax to classify each RoI and a regression layer to fine-tune the bounding box coordinates.

### 7.5 Post-processing

- Apply non-maximum suppression (NMS) to remove redundant overlapping bounding boxes.
- Finalize the detection results with class labels and refined bounding boxes.

## 8 Datasets and Evaluation

### 8.1 Datasets

For object detection, a subset of the COCO dataset is commonly used:

- **COCO Dataset:** COCO Dataset (Subset) provides diverse images with bounding box annotations.

### 8.2 Evaluation Metrics

- **Mean Average Precision (mAP):** Evaluates the detection performance over multiple classes.
- **Intersection over Union (IoU):** Assesses the quality of predicted bounding boxes.