Computer Vision

T5 Bootcamp by SDAIA

Agenda

- Segmentation
- Semantic Segmentation
- Instance Segmentation
- Semantic Segmentation Models
- Instance Segmentation Models
- Pre-Trained Segmentation Models

Segmentation

What is Segmentation?

Segmentation is a fundamental computer vision task that divides digital images into segments, also known as pixel sets.

The aim is to make an image simpler and easier to understand and analyze by changing its representation.

Image Segmentation Techniques

Edge-Based Segmentation

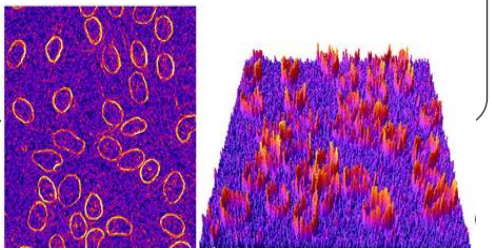
Definition: Identifies object edges in an image.
Features: Utilizes contrast, texture, color, and saturation variations to locate edges.
Benefits: Reduces image size, facilitates analysis, accurately represents object borders.

Threshold-Based Segmentation

Definition: Divides pixels based on intensity relative to a given threshold.
Usage: Suitable for objects with higher intensity than backgrounds.
Process: Produces a binary image by dividing a grayscale image into segments based on threshold value.

Region-Based Segmentation

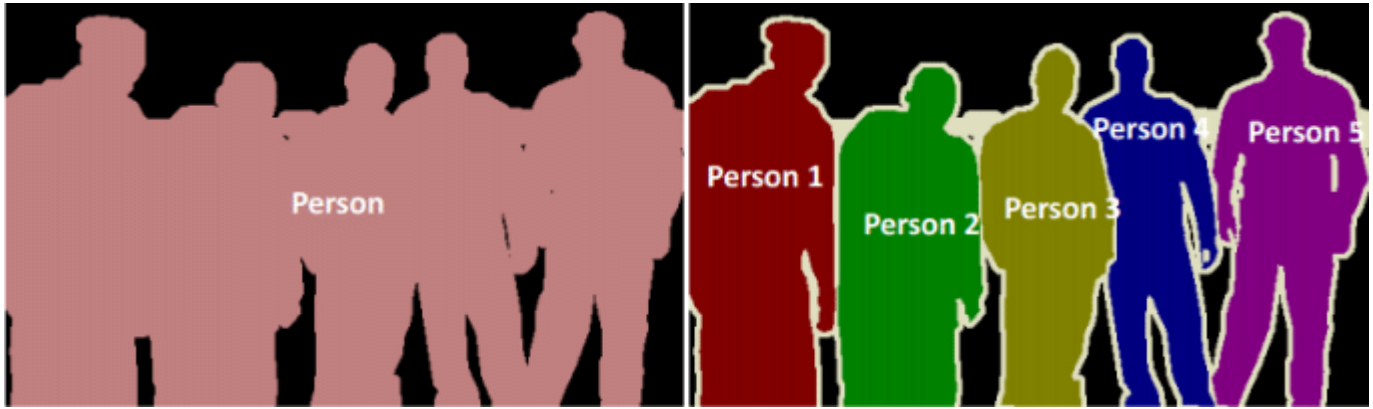
Definition: Divides an image into regions with similar characteristics.
Procedure: Locates regions using seed points and grows or shrinks them based on neighboring pixels.
Advantages: Enables grouping of pixels into meaningful regions



Types of Image Segmentation

There are various ways to segment an image. Here are some of the main techniques:

- **Semantic Segmentation:** Semantic segmentation involves arranging the pixels in an image based on semantic classes.
- **Instance Segmentation:** Instance segmentation involves classifying pixels based on the instances of an object *(as opposed to object classes)*.



Semantic Segmentation

Instance Segmentation

Instance Segmentation is a unique form of image segmentation that deals with detecting and delineating each distinct instance of an object appearing in an image.

Semantic segmentation aims to extract features before using them to form distinct categories in an image. The steps involved are as follows:

- Analyze training data for classifying a specific object in the image.
- Create a semantic segmentation network to localize the objects and draw a bounding box around them.
- Train the semantic segmentation network to group the pixels in a localized image by creating a segmentation mask.

Image Recognition vs Semantic Segmentation

The steps in semantic segmentation differ significantly from image classification, where we only assign a single class to the whole image.

- **Image Recognition**: Identifies objects or scenes within an image without distinguishing between individual pixels, providing a holistic understanding of the image content.
- **Semantic Segmentation**: Assigns labels to each pixel, enabling detailed understanding of object boundaries and contexts within the image.

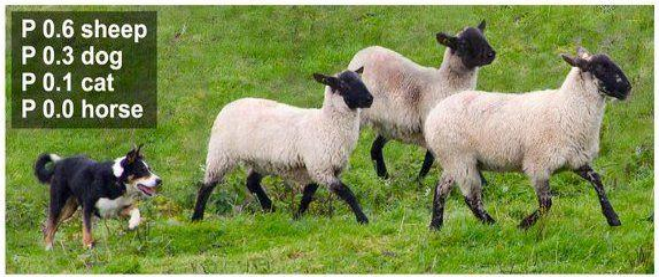
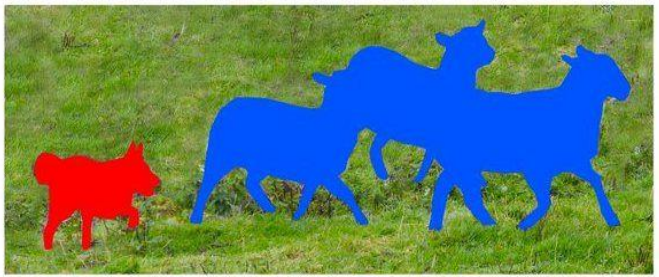


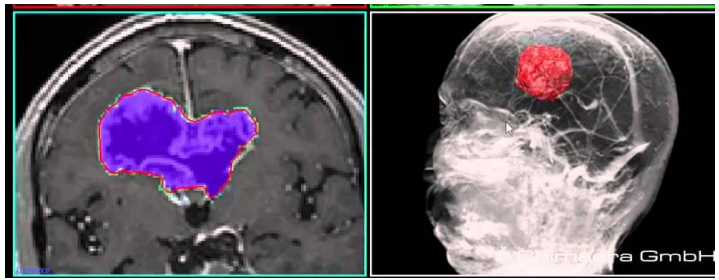
Image Recognition



Semantic Segmentation

Applications of Semantic Segmentation

- **Medical Diagnostics:** For detecting medical abnormalities in X-Rays, CT Scans, MRI Scans
- **Geo-Sensing**: For land usage mapping from satellite imagery and monitoring areas of deforestation and urbanization
- **Autonomous Driving:** For accurately detecting lanes, pedestrians, traffic signs, road, sky and other vehicles on the road.



Instance Segmentation

Instance Segmentation

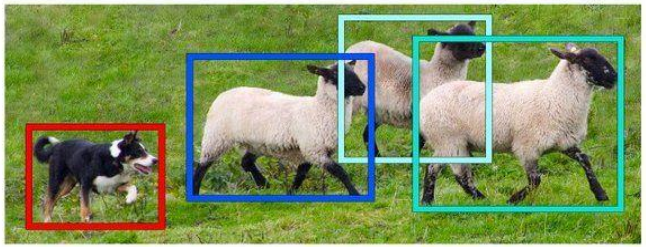
Instance Segmentation is a unique form of image segmentation that deals with detecting and delineating each distinct instance of an object appearing in an image.

Instance segmentation detects all instances of a class with the extra functionality of separating instances of any segment class.

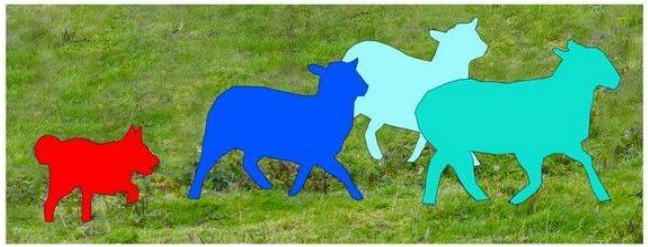
How does Instance Segmentation work?

Instance segmentation involves identifying boundaries of the objects at the detailed pixel level, making it a complex task to perform. Instance segmentation contains 2 significant parts:

- **Object Detection**: Firstly, it runs object detection to find all bounding boxes for every object in an image
- **Semantic Segmentation**: After finding all the rectangles (bounding boxes), it uses a semantic segmentation model inside every rectangle



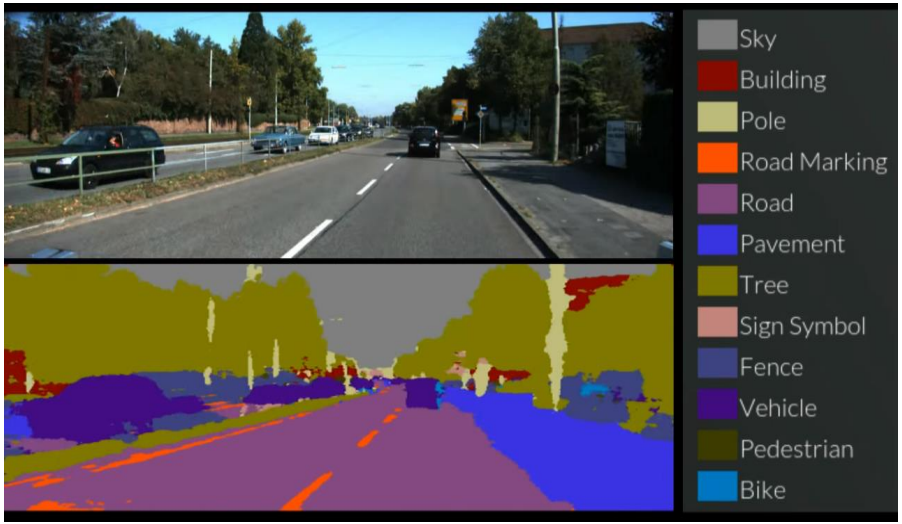
Object Detection



Instance Segmentation

Applications of Instance Segmentation

- **Self-Driving Cars:** Used in conjunction with dense distance to object estimation methods to provide high-resolution 3D depth estimation of a scene from monocular 2D images
- **Robotics:** Used with self-supervised learning to segment visual observations into individual objects by interacting with the environment
- **Automation:** Used for detecting dents on a car, separating buildings in a city, and more.



Semantic vs. Instance Segmentation

Semantic Segmentation	Instance Segmentation
For each pixel, it detects the object category it belongs to. where all object labels are known to the model	For each pixel, it identifies the object instance it belongs to. It differentiates two objects with the same labels.
Firstly, target detection takes place, and then each pixel is labelled.	It is a hybrid of annotation of target detection and semantic segmentation.
Open-source datasets: • Stanford Background Dataset • Microsoft COCO Dataset • KITTI Dataset	Open-source datasets: • LiDAR Bonnetal Dataset • HRSID (High-Dimension SAR Images Dataset) • Pascal SBD Dataset

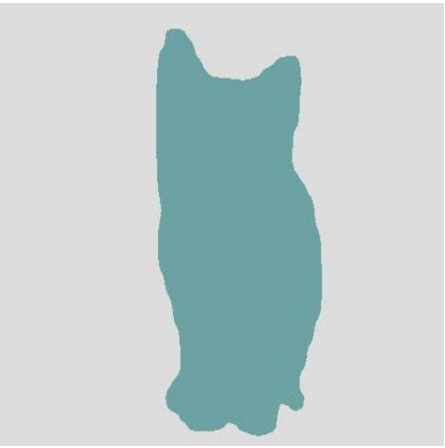
Semantic Segmentation Models

Semantic Segmentation Modeling

The Semantic Segmentation models try to find answers to two questions:

- **“What” is in the input image?**
  - **Ans**: Cat and Background
- **“Where” is that object in the input image?**
  - **Ans**: The location of the cat in the image.

Input of Image Segmentation

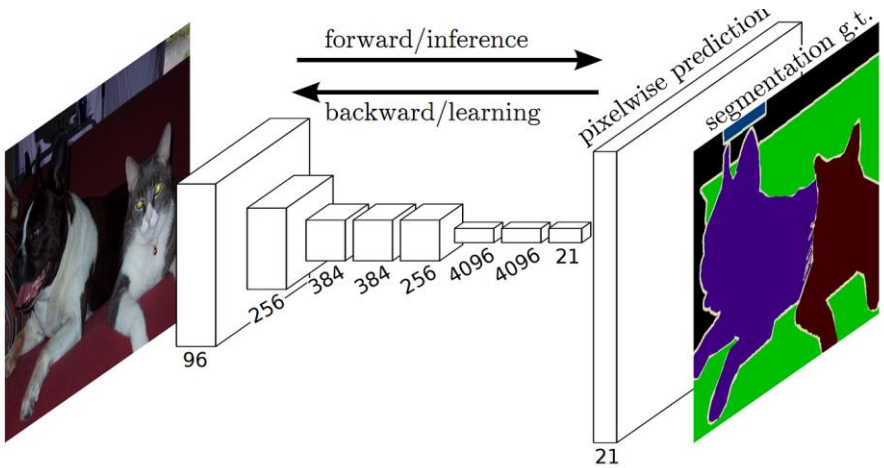


Output for Image Segmentation

Models for Semantic Segmentation

Many different algorithms and techniques exist for semantic segmentation. Some of the most commonly used ones include:

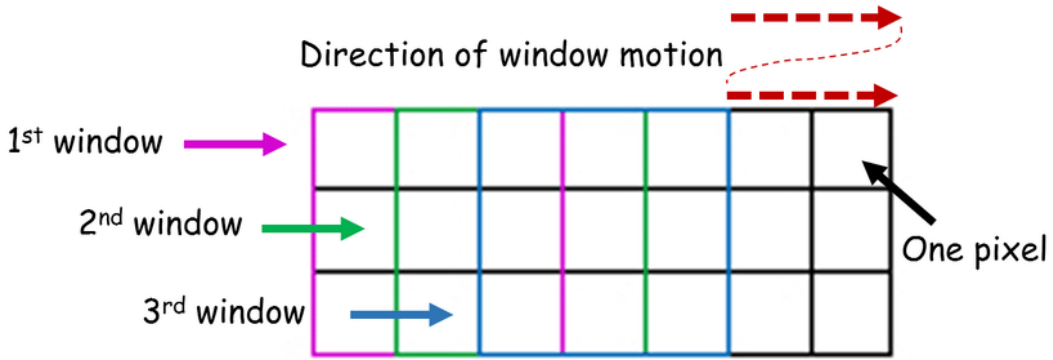
- **U-Net:** U-Net architecture has a contracting path to capture context and a symmetric expanding path for precise localization.



A fully convolutional image segmentation network
Photo Credit : https://arxiv.org/pdf/2001.05566.pdf

U-Net

- Olaf Ronneberger and his team developed U-Net in 2015 for their work on biomedical images.
- It won the ISBI challenge by outperforming the sliding window technique by using fewer images and data augmentation to increase the model performance.
- Sliding window architecture performs localization tasks well on any given training dataset.
- This architecture creates a local patch for each pixel, generating separate class labels for each pixel.

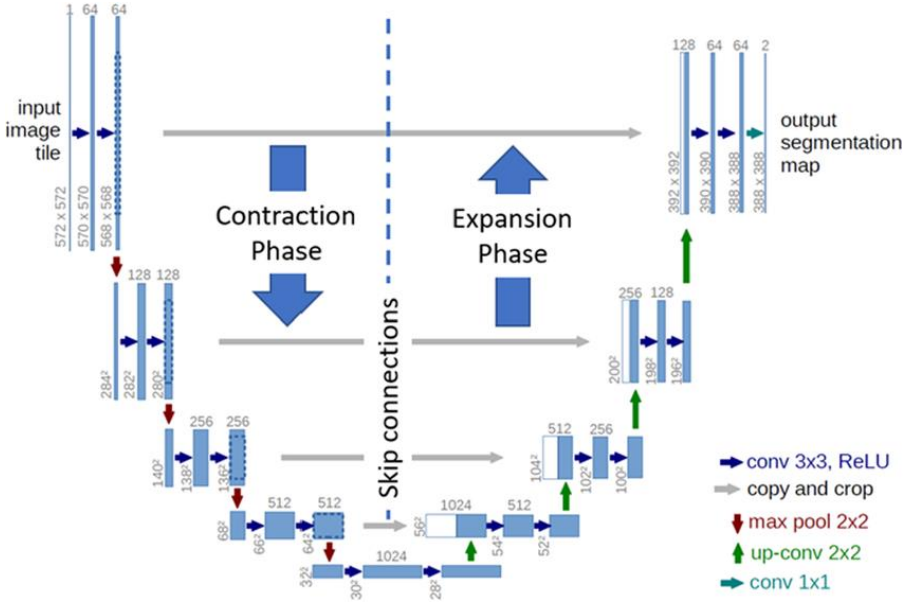


U-Net Architecture

U-Net gets its name from its architecture. The "**U**" shaped model comprises convolutional layers and two networks.

First is the Contraction Phase, which is followed by the Expansion Phase.

With the U-Net, we can solve the previous two questions of segmentation: "**What** is in the input image?" and "**where** is that object in the input image?"

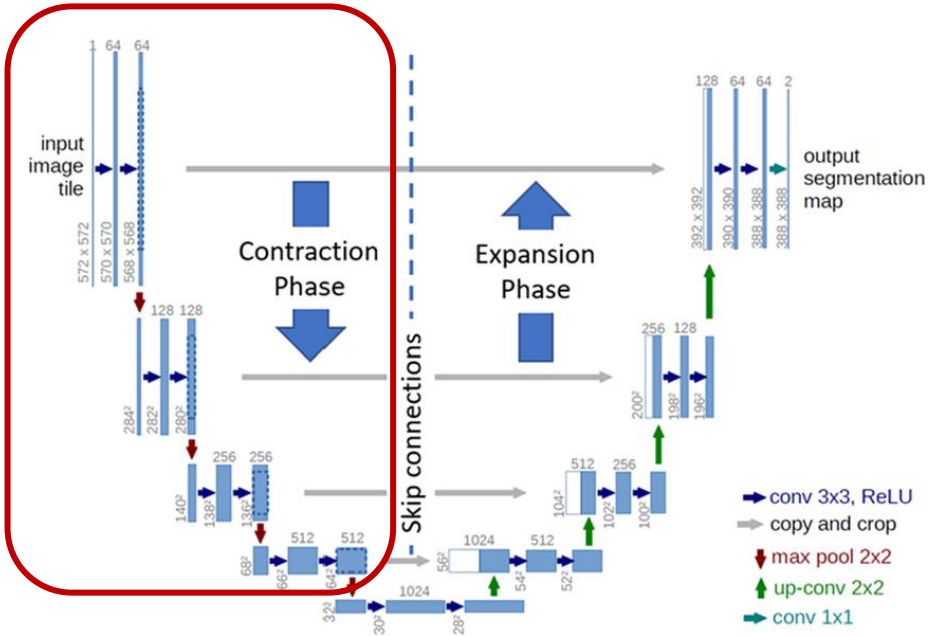


U-net architecture (Ronneberger et al., 2015)

U-Net Architecture: **Contracting Network**

Contracting Network (Encoder Network):

- Learns feature map of input image.
- Addresses "**what**" is in the image.
- Contains 4 contracting blocks with two 3x3 convolutional layers and ReLU activation.
- Followed by max pooling layer to reduce spatial dimensions.
- No fully connected layers in the end, as the output required now is not the class label but a mask of the same size as our input image.

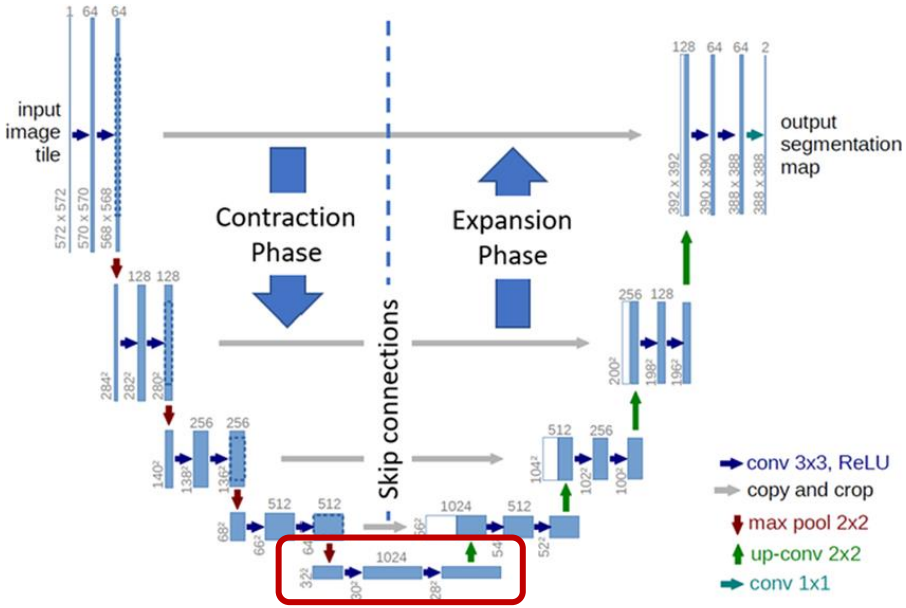


U-net architecture (Ronneberger et al., 2015)

U-Net Architecture: **Bottleneck**

Bottleneck Layer:

- Positioned between Contracting and Expansion networks.
- Consists of 2 convolutional layers with ReLU activation.
- Produces final feature map representation.

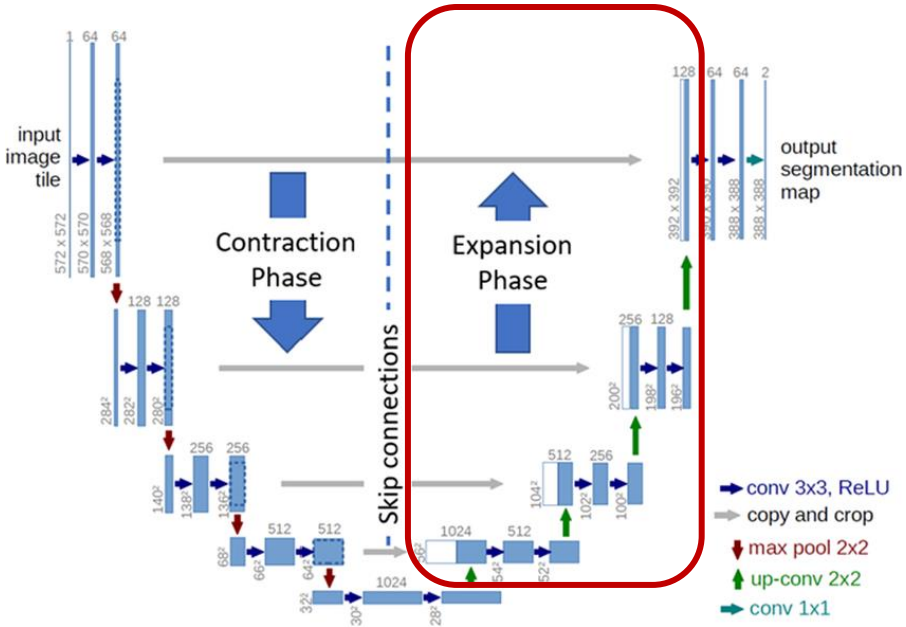


U-net architecture (Ronneberger et al., 2015)

U-Net Architecture: **Expansive Network**

Expansive Network (Decoder Network):

- Upsamples feature maps to match input image size.
- Generates segmentation mask using skip connections.
- Addresses "**where**" is the object in the image.
- Comprises 4 expansive blocks, each starting with a transpose convolution (up-conv).
- Utilizes two 3x3 convolutional layers with ReLU activation after concatenating output with corresponding skip layer connection.

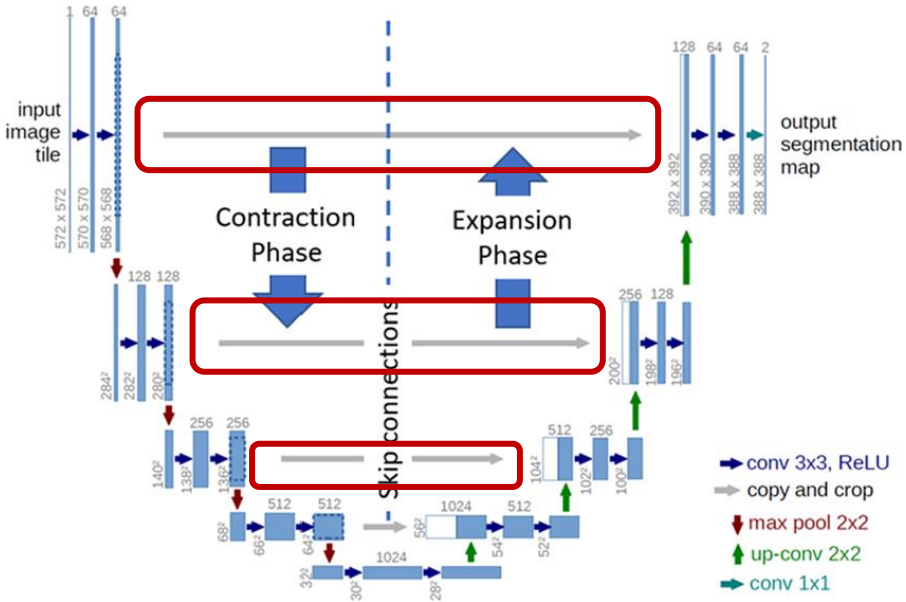


U-net architecture (Ronneberger et al., 2015)

U-Net Architecture: **Skip Connections**

Skip Connections:

- Represented by grey arrows in the model architecture.
- Utilizes contextual feature information from encoder blocks to generate segmentation map.
- Enables projection of high-resolution features learned from contraction blocks to the feature map output from the bottleneck layer.



U-net architecture (Ronneberger et al., 2015)

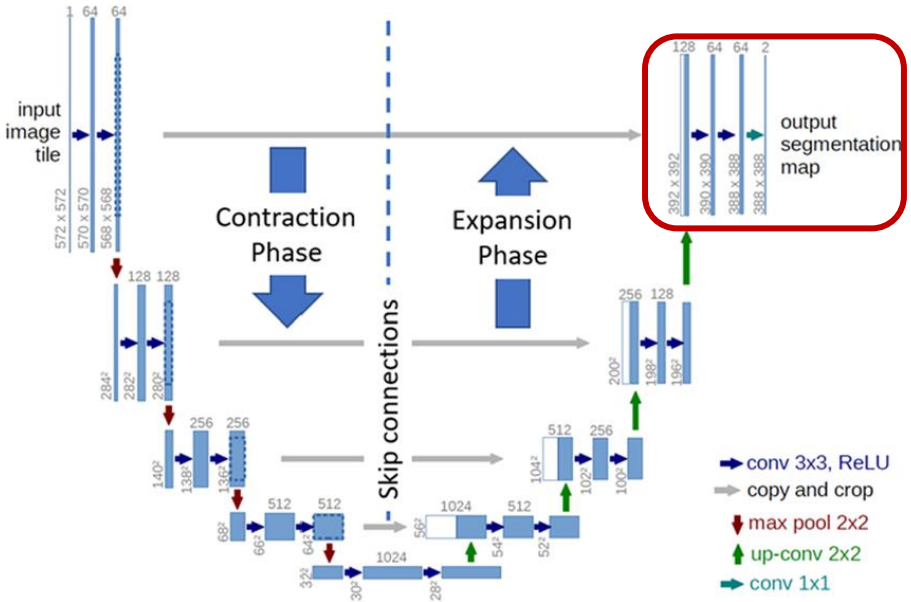
U-Net Architecture: **Segmentation Mask**

Final Segmentation Mask:

- Follows the last Expansive block, comprising a 1x1 convolution with sigmoid activation.
- Outputs segmentation mask containing pixel-wise classification.

Contracting and Expansive Paths Interaction:

- Contracting path passes information to expansive path.
- Allows capturing both feature information and localization.



U-net architecture (Ronneberger et al., 2015)

U-Net Implementation Code

```
import tensorflow as tf
from tensorflow.keras import layers, models

def unet_model(input_shape):
    inputs = tf.keras.Input(shape=input_shape)

    # Downsample blocks
    conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(inputs)
    conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(conv1)
    pool1 = layers.MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = layers.Conv2D(128, 3, activation='relu', padding='same')(pool1)
    conv2 = layers.Conv2D(128, 3, activation='relu', padding='same')(conv2)
    pool2 = layers.MaxPooling2D(pool_size=(2, 2))(conv2)

    # Bottleneck
    conv3 = layers.Conv2D(256, 3, activation='relu', padding='same')(pool2)
    conv3 = layers.Conv2D(256, 3, activation='relu', padding='same')(conv3)
```


U-Net Implementation Code *(cont'd)*

```
    # Upsample blocks
    up4 = layers.Conv2DTranspose(128, 2, strides=(2, 2), padding='same')(conv3)
    up4 = layers.concatenate([up4, conv2], axis=3)
    conv4 = layers.Conv2D(128, 3, activation='relu', padding='same')(up4)
    conv4 = layers.Conv2D(128, 3, activation='relu', padding='same')(conv4)

    up5 = layers.Conv2DTranspose(64, 2, strides=(2, 2), padding='same')(conv4)
    up5 = layers.concatenate([up5, conv1], axis=3)
    conv5 = layers.Conv2D(64, 3, activation='relu', padding='same')(up5)
    conv5 = layers.Conv2D(64, 3, activation='relu', padding='same')(conv5)

    # Output layer
    outputs = layers.Conv2D(1, 1, activation='sigmoid')(conv5)
    model = tf.keras.Model(inputs=inputs, outputs=outputs, name='unet_model')
    return model

# Example usage
input_shape = (256, 256, 3)
model = unet_model(input_shape)
model.summary()
```


Evaluation Metrics

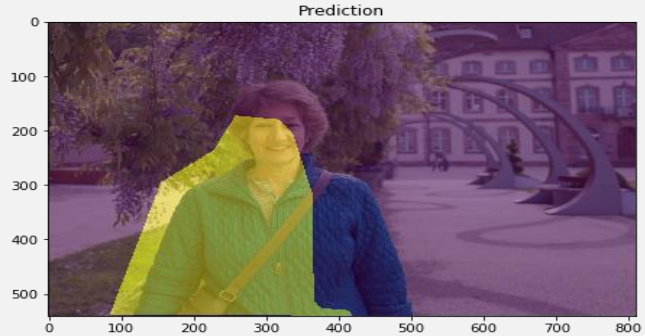
To evaluate the performance of a model we need metrics that evaluate them. The evaluation metrics for U-Net (Semantic Segmentation) model are:

Pixel Accuracy:

- Measures the proportion of correctly classified pixels in the generated segmentation mask.
- Simplistic metric but may not fully assess model performance.
- Prone to bias in cases of extreme class imbalance in the dataset.

Intersection Over Union (IoU):

- Calculates the area of overlap between predicted and ground truth segmentations divided by the area of their union.
- Ranges from 0 to 1, where 0 indicates no overlap and 1 indicates perfect overlap.
- Provides a comprehensive assessment of segmentation quality, addressing potential biases in pixel accuracy.



Exercise

Transitioning to Google-Colab for hands-on coding practice.

Notebook:

- TBD

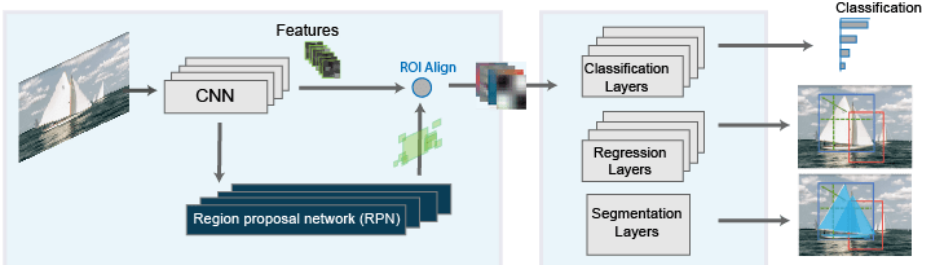
Instance Segmentation Models

Instance Segmentation Modeling

Instance segmentation is more complex because the model identifies each object instance. It combines the tasks of object detection and semantic segmentation.

Although it can be very different depending on the application, the process generally involves:

1. **Object Detection**: The model identifies bounding boxes around each object instance.
2. **Pixel Classification**: Similar to semantic segmentation, each pixel within the bounding box is categorized.
3. **Instance Differentiation**: The model distinguishes between different instances of the same category within the image.



Instance Segmentation Tasks

The Instance Segmentation Modeling Key functionalities are:

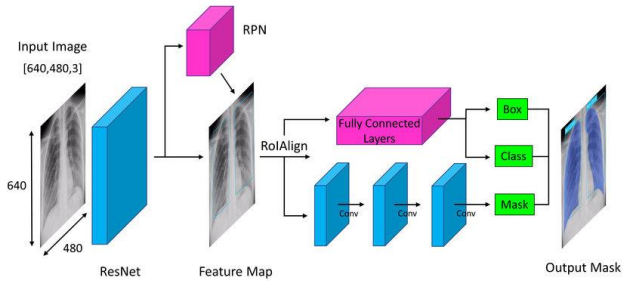
- **Classification**
  - **Q:** What are in the image?
- **Localization**
  - **Q:** Where are they?
- **Mask (per pixel) classification - Where+ ?**
  - **Ans**: More precise to bounding box
- **Landmarks localization - What+, Where+ ?**
  - **Ans**: Not only per-pixel mask, but also key points in the objects

Models for Instance Segmentation

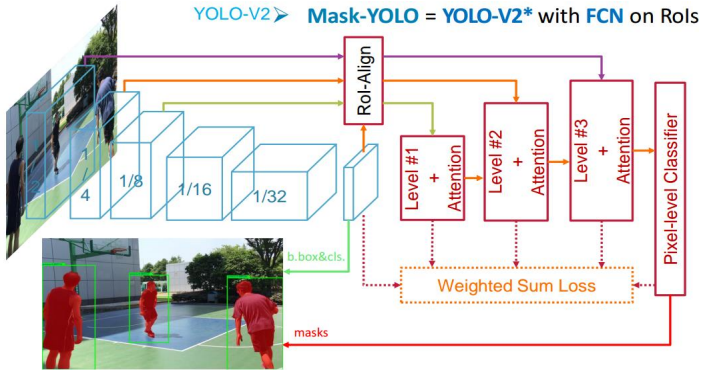
Similar to semantic segmentation, several models excel at Instance Segmentation tasks.

Some of the most commonly used ones include:

- **Mask R-CNN**: An extension of Faster R-CNN, this model adds a branch for predicting segmentation masks on each Region of Interest (RoI). This effectively combines object detection with pixel-wise segmentation.



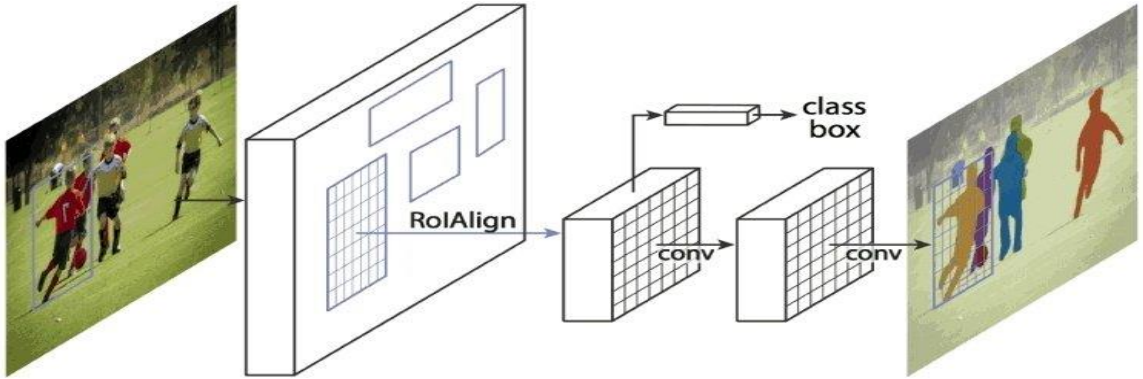
- **YOLO (You Only Look Once)**: Known for their speed, some open-sourced YOLO versions adapt to perform instance segmentation by adding segmentation capabilities.



Mask R-CNN

Mask R-CNN, or Mask-RCNN, is a Convolutional Neural Network (CNN) and state-of-the-art in terms of image segmentation and instance segmentation.

Mask R-CNN was developed on top of Faster R-CNN, a Region-Based Convolutional Neural Network**.**

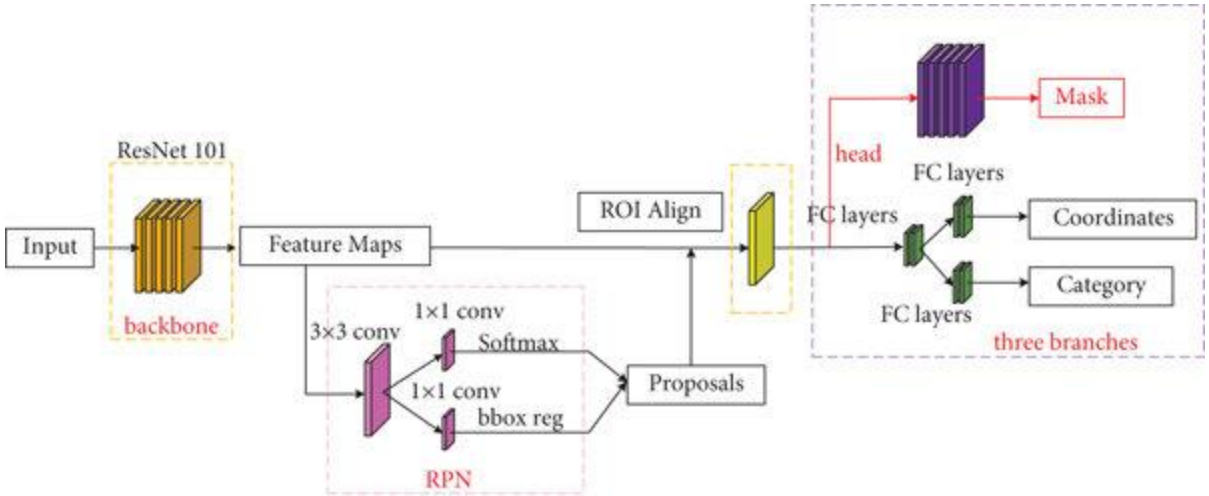


Mask R-CNN Framework for Instance Segmentation

How does Mask R-CNN work?

Mask R-CNN was built using Faster R-CNN. While Faster R-CNN has 2 outputs for each candidate object, a class label, and a bounding-box offset, **Mask R-CNN is the addition of a third branch that outputs the object mask**.

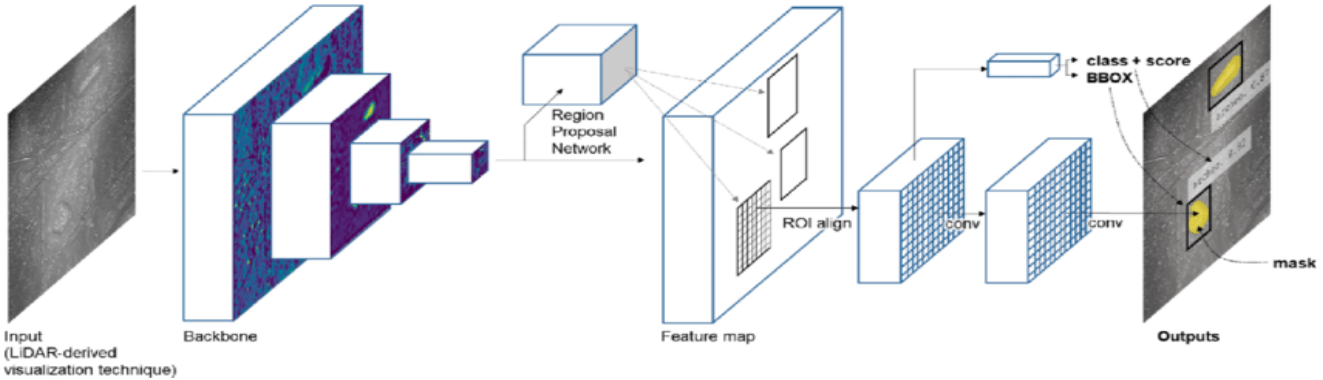
- The additional mask output is distinct from the class and box outputs, requiring the extraction of a much finer spatial layout of an object. Mask R-CNN works by adding a branch for predicting an object mask (Region of Interest) in parallel with the existing branch for bounding box recognition.



Mask R-CNN Architecture

Let's first quickly understand the intuition behind Mask R-CNN:

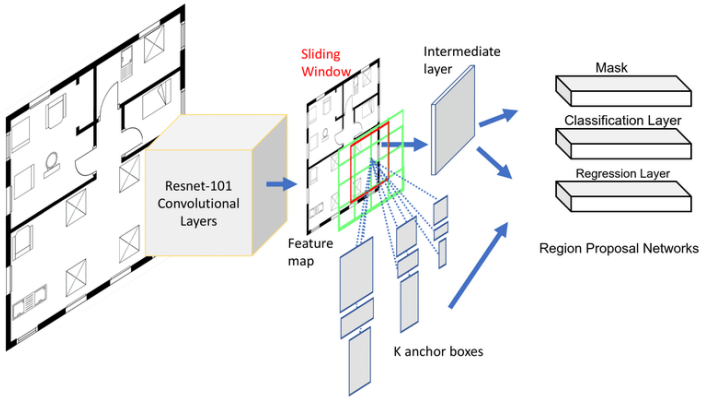
1. Faster R-CNN first uses a ConvNet to extract feature maps from the images
2. These feature maps are then passed through a Region Proposal Network (RPN) which returns the candidate bounding boxes
3. Then apply an RoI pooling layer on these candidate bounding boxes to bring all the candidates to the same size
4. Finally, the proposals are passed to a fully connected layer to classify and output the bounding boxes for objects then Segmentation Mask is applied

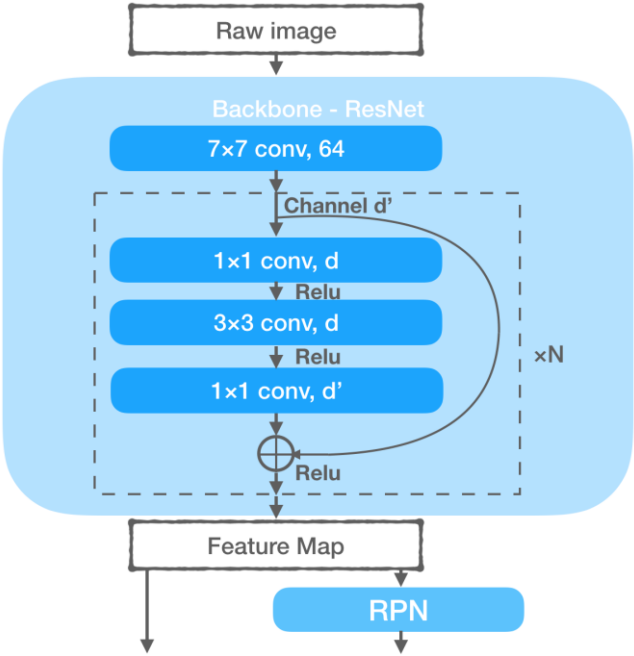


Mask R-CNN: **Backbone Model**

Mask-RCNN employs the ResNet 101 architecture, similar to the ConvNets utilized in Faster R-CNN, to extract feature maps from images.

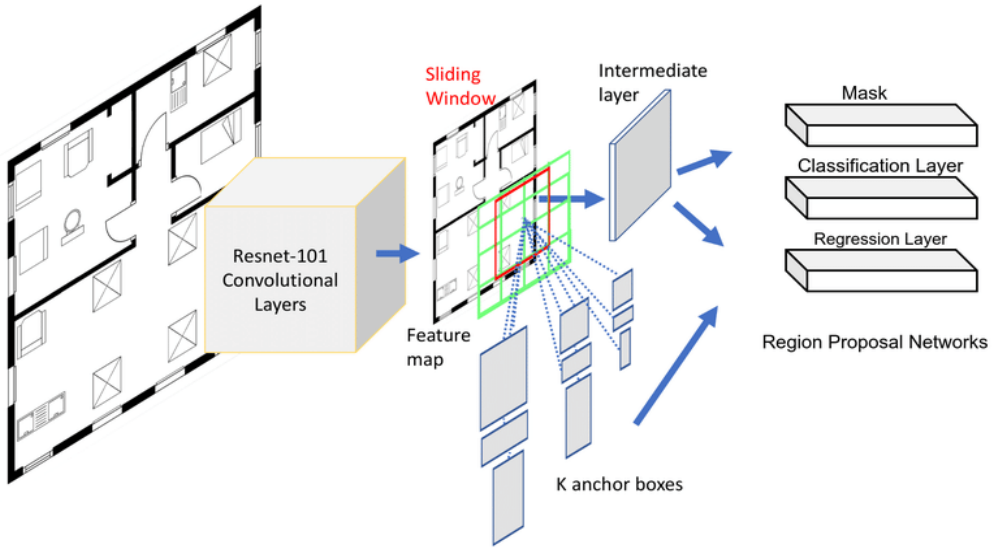
So, the first step is to take an image and extract features using the ResNet 101 architecture.





Mask R-CNN: **Region Proposal Network (RPN)**

- The feature maps obtained in the previous step and apply a region proposal network (RPM).
- This basically predicts if an object is present in that region (or not). In this step, we get those regions or feature maps which the model predicts contain some object.



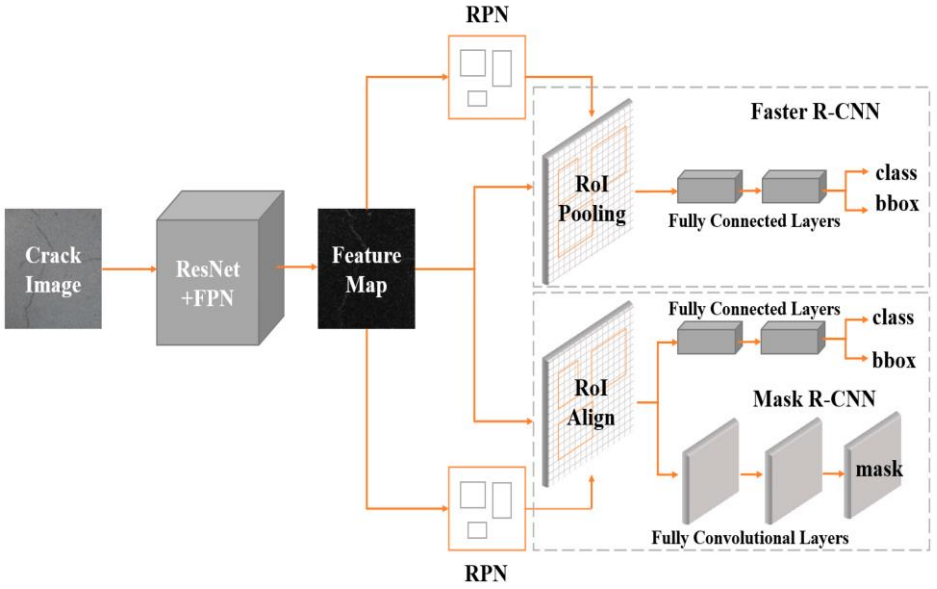
Mask R-CNN: **Region of Interest (RoI)**

ROI utilizes Region Proposal Network (RPN) to identify regions of interest with varying shapes, subsequently standardizing them through a pooling layer, followed by a fully connected network for accurate class label and bounding box prediction.

- Mask R-CNN similar to Faster R-CNN until this point.

Faster R-CNN vs Mask R-CNN:

- Mask R-CNN additionally generates segmentation masks, setting it apart from Faster R-CNN.
- Evaluates IoU with ground truth boxes, considering regions with IoU ≥ 0.5 as regions of interest.



ROI in Faster R-CNN vs Mask R-CNN

Mask R-CNN: **Segmentation Mask**

- Mask branch added to architecture to generate segmentation masks for each region containing an object.
- Masks returned are of size 28x28 for each region which is then scaled up for inference.

Advantages & Disadvantages of Mask R-CNN

PROS

- ✓ **Simplicity**: Mask R-CNN is simple to train.
- ✓ **Performance**: Mask R-CNN outperforms all existing, single-model entries on every task.
- ✓ **Efficiency**: The method is very efficient and adds only a small overhead to Faster R-CNN.

CONS

- ✓ **Complexity**: Implementing and fine-tuning Mask R-CNN requires expertise and computational resources due to its complexity.
- ✓ **Speed**: Multi-stage architecture results in slower inference times.
- ✓ **Training Data Requirements**: Effective training needs large annotated datasets, which may be difficult or costly to obtain.

Exercise

Transitioning to Google-Colab for hands-on coding practice.

Notebook:

- TBD

Pre-Trained Segmentation Models

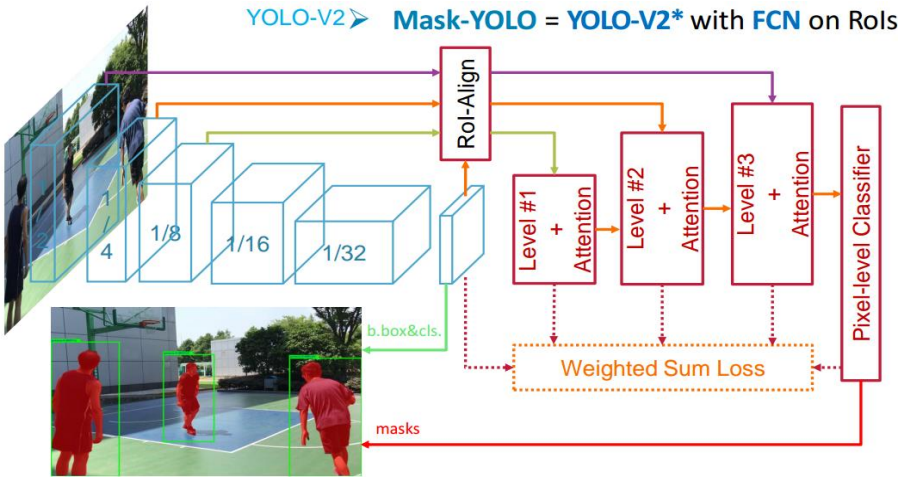
YOLO for Instance Segmentation

YOLO (You Only Look Once) is a popular object detection algorithm known for its speed and accuracy.

While traditionally used for object detection, different versions of YOLO can be adapted and utilized for instance segmentation tasks.

Adaptability:

- YOLO architecture can be modified and extended to perform instance segmentation, where the goal is to not only detect objects but also segment them at the pixel level.
- Different versions of YOLO, such as *YOLOv1, YOLOv2 (YOLO9000), YOLOv3, and YOLOv4, etc.*. can be adapted and fine-tuned for instance segmentation tasks.



YOLO's flexibility

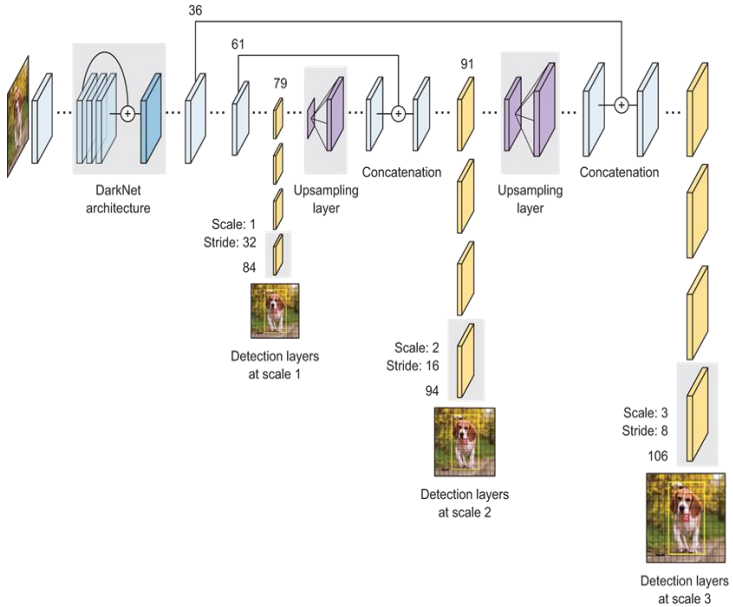
In instance segmentation, YOLO is trained to not only predict bounding boxes around objects but also generate segmentation masks that precisely delineate object boundaries.

By incorporating additional layers or modifying the network architecture, YOLO can output pixel-wise segmentation masks alongside object detection predictions

YOLO's flexibility allows researchers and practitioners to experiment with different YOLO versions and configurations to achieve optimal performance for instance segmentation tasks.

Recap: YOLO for Object Detection

- **Single Shot Approach**: YOLO (You Only Look Once) revolutionizes object detection by adopting a single-shot approach, eliminating the need for multiple passes through the image.
- **CNNs:** YOLO harnesses the power of CNNs to process the entire image at once, enabling rapid and efficient detection of objects in real-time.
- **Grid-Based Prediction**: The input image is divided into a grid, and each grid cell predicts multiple bounding boxes and their corresponding class probabilities simultaneously.
- **Feature Extraction:** CNN layers extract high-level features from the input image, capturing spatial information crucial for object detection.
- **Efficient Inference:** YOLO's unified architecture enables fast and accurate inference by directly predicting bounding boxes and class probabilities in a single pass, without the need for complex post-processing steps.

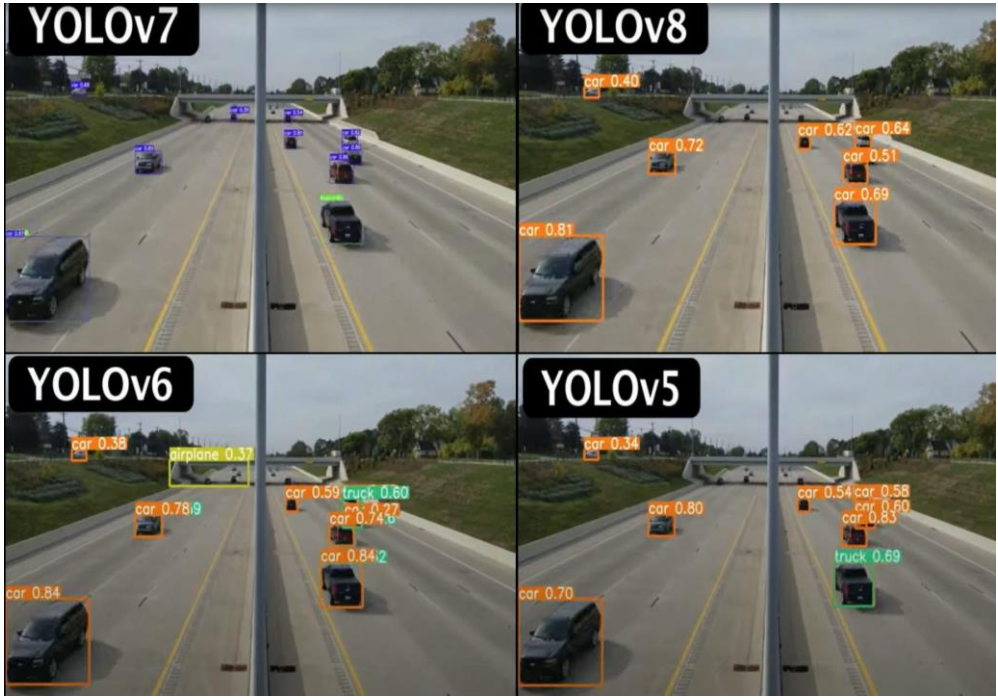


YOLOv3 network architecture

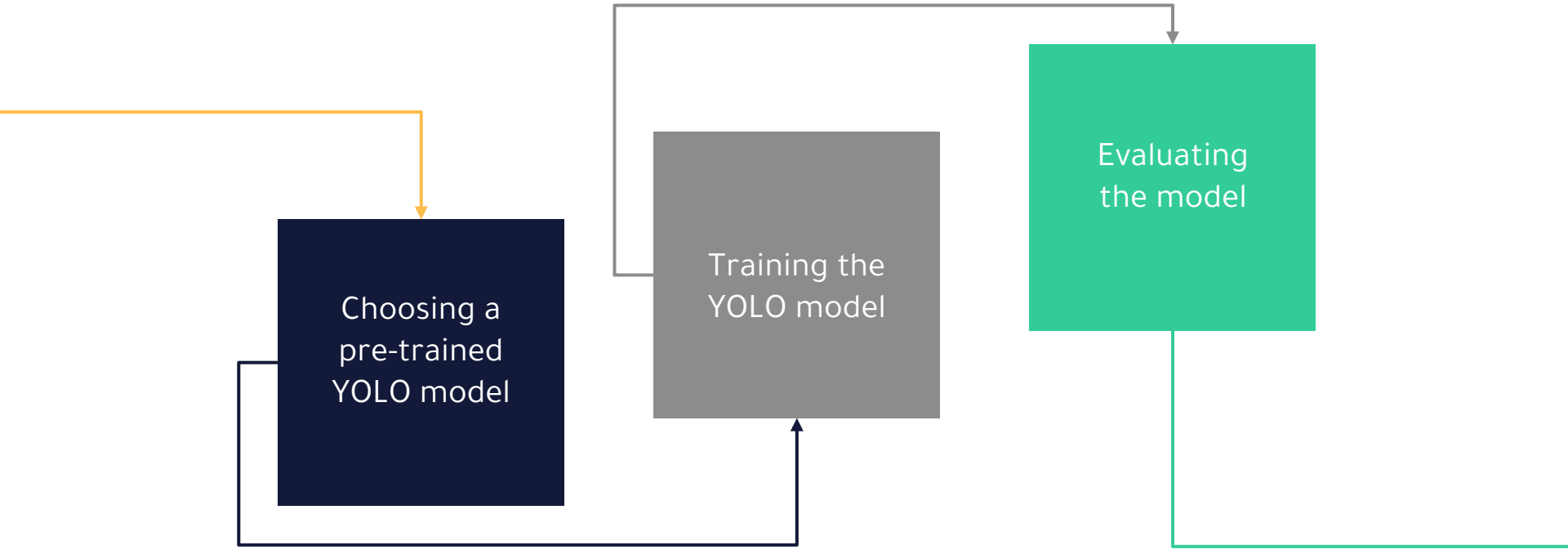
Why use YOLO for Segmentation?

There are various benefits to using YOLO for segmentation. YOLO models are speedy and ideal for real-time usage:

- ✓ **Speed**: YOLO models are very fast, making them suitable for real-time applications.
- ✓ **Accuracy**: YOLO models are also very accurate, achieving state-of-the-art results on many segmentation benchmarks.
- ✓ **Robustness**: YOLO models are robust to noise and occlusions, making them suitable for challenging real-world environments.

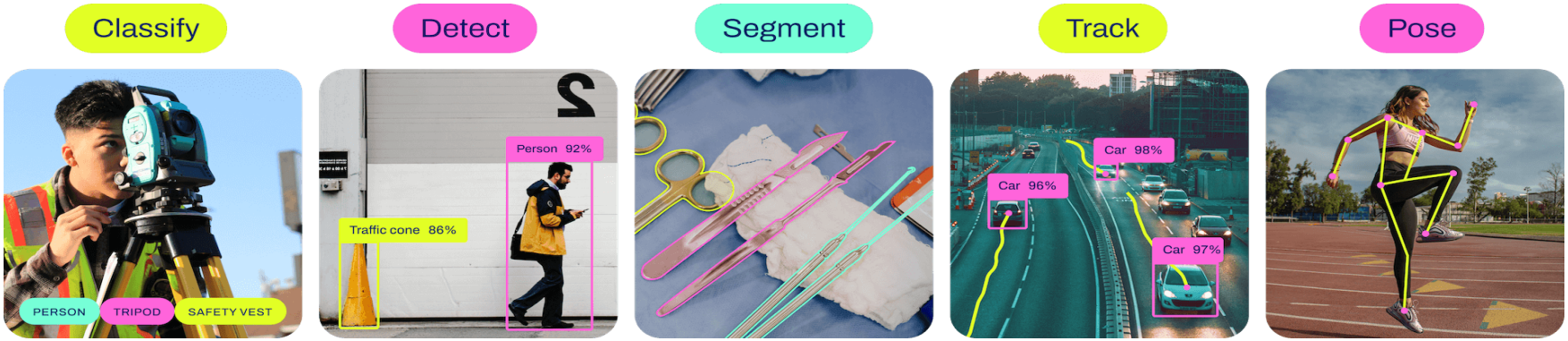


YOLO for Segmentation Practical Applications



Pre-Trained YOLOv8

- YOLOv8 is a state-of-the-art object detection algorithm renowned for its speed and accuracy.
- Pre-trained YOLOv8 models are available for various segmentation tasks, including detection, segmentation, tracking, and pose estimation.
- YOLOv8 pretrained Segment models (Detect, Segment, Track and Pose models) are pretrained on the COCO dataset, while YOLO Classify models are pretrained on the ImageNet dataset.



Segmentation Pre-Trained Datasets

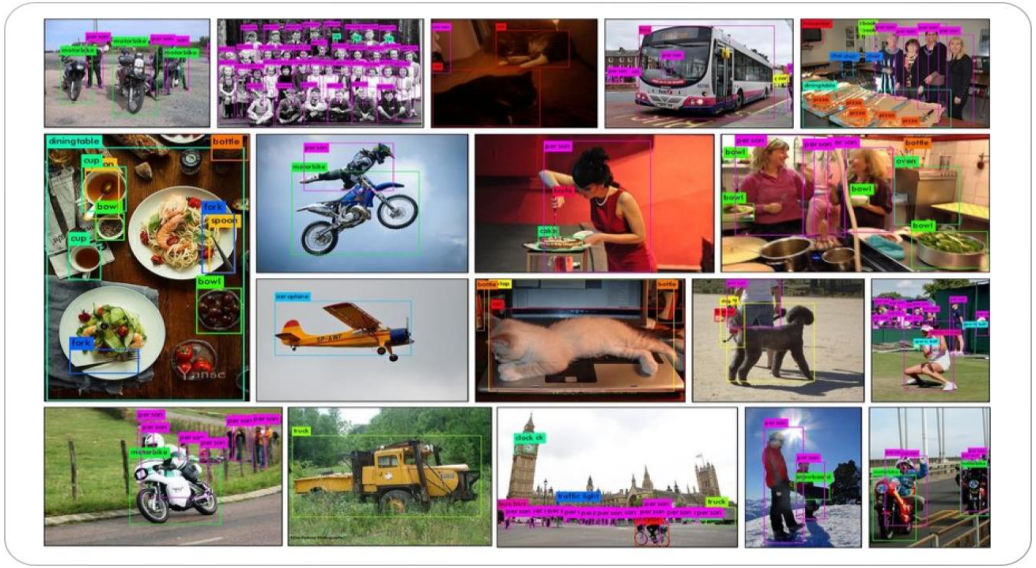
YOLO models are often pre-trained on large-scale datasets such as:

- **COCO (Common Objects in Context)**: Contains diverse images with 80 object categories, annotated with bounding boxes and segmentation masks.
- **VOC (Visual Object Classes):** Includes images with 20 object categories, annotated for object detection, segmentation, and classification tasks.
- **ImageNet:** Known for its vast collection of images across numerous categories, often used for pre-training CNNs for object recognition tasks
- **Open Images Dataset**: A massive dataset with millions of images across various categories, annotated for object detection and segmentation.

COCO Dataset (Common Objects in Context)

The COCO (Common Objects in Context) dataset is a comprehensive collection used for object detection, segmentation, and captioning tasks in computer vision research.

It encompasses over 330,000 images, each meticulously annotated with 80 object categories and accompanied by 5 descriptive captions, making it a valuable resource for various computer vision applications.



COCO Dataset (cont'd)

Dataset Structure:

- Images organized into train, validation, and test sets within a hierarchical directory structure.
- Annotations provided in JSON format, with each file corresponding to a single image.
- Each annotation includes detailed information such as object class, bounding box coordinates, segmentation masks, keypoints, and descriptive captions.

Annotation Details:

- Image file name and size (width and height).
- Object details: class, bounding box coordinates, segmentation mask, keypoints (if available).
- Captions describing the scene.

COCO Dataset (cont'd)

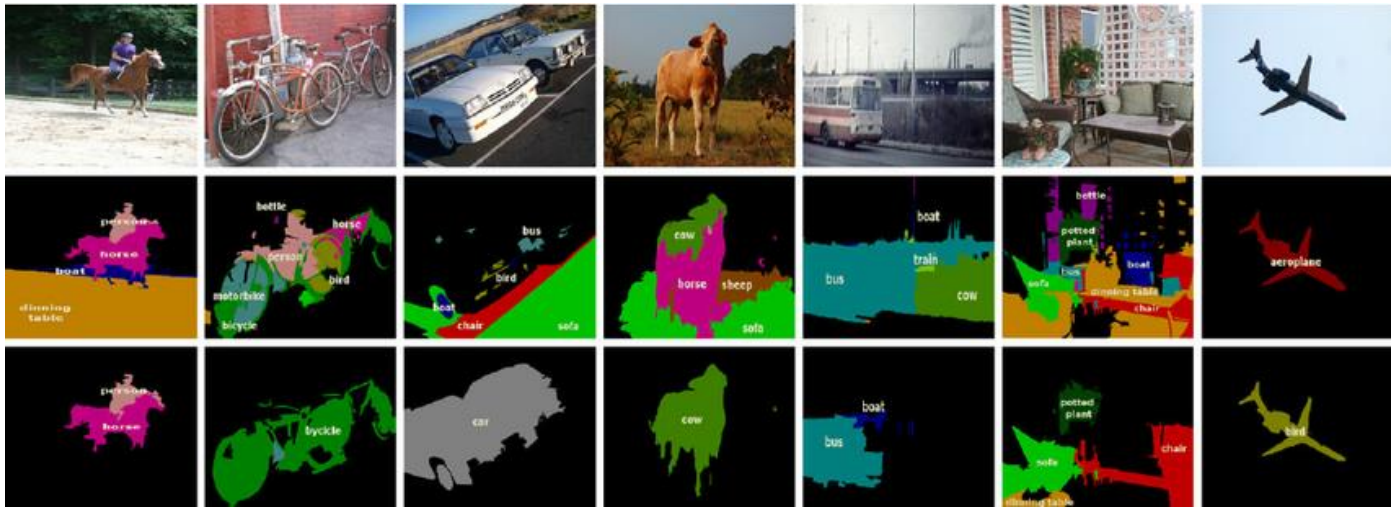
Types of Annotations:

- **Object detection**: Bounding box coordinates and segmentation masks for 80 different object classes.
- **Stuff image segmentation**: Pixel maps for 91 background areas.
- **Panoptic segmentation**: Identifies both "things" and "stuff" categories.
- **Dense pose:** Mapping between pixels and a template 3D model.
- **Keypoint annotations**: Over 250,000 persons annotated with key points.

VOC (Visual Object Classes)

The PASCAL VOC (Visual Object Classes) dataset is a renowned benchmark dataset for object detection, segmentation, and classification tasks in computer vision research.

It facilitates research on various object categories and serves as a standard benchmark for evaluating computer vision models..



VOC Dataset (cont'd)

Dataset Structure:

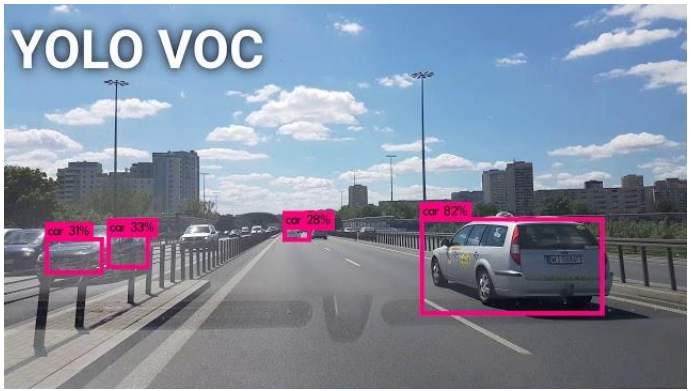
- Includes two main challenges: VOC2007 and VOC2012.
- Consists of 20 object categories, covering common objects like cars, bicycles, animals, as well as specific categories such as boats, sofas, and dining tables.
- Annotations comprise object bounding boxes, class labels, and segmentation masks, supporting diverse tasks in object detection, segmentation, and classification.
- Provides standardized evaluation metrics like mean Average Precision (mAP) for assessing model performance, enabling fair comparison among different approaches.

	train		val		trainval		test	
	Images	Objects	Images	Objects	Images	Objects	Images	Objects
VOC 2007	2501	6301	2510	6307	5011	12608	4952	12032
VOC 2012	5717	13609	5823	13841	11540	27450	11540	27450

VOC Dataset (cont'd)

Applications:

- Widely utilized for training and evaluating deep learning models in object detection (e.g., YOLO, Faster R-CNN, SSD), instance segmentation (e.g., Mask R-CNN), and image classification.
- Offers a diverse set of object categories, extensive annotated images, and standardized evaluation metrics, making it indispensable for computer vision research and applications.



ImageNet-S

The ImageNet-S dataset, derived from ImageNet, is a variant that focuses on semantic segmentation, providing pixel-level annotations for a subset of the original images.

- ImageNet initially designed for image classification, ImageNet-S has emerged as a valuable resource for training segmentation models due to its pixel-level annotations.

Dataset Overview:

- With annotations for object boundaries and regions, ImageNet-S facilitates training and evaluation of segmentation models at a finer granularity.

ImageNet-S *(cont'd)*

While ImageNet-S inherits the diverse imagery of ImageNet, it augments it with pixel-wise annotations, making it suitable for segmentation tasks.

- Researchers leverage ImageNet-S for training segmentation models, utilizing both the rich feature representations learned from ImageNet and the detailed segmentation annotations.

Fine-tuning Pre-trained Models:

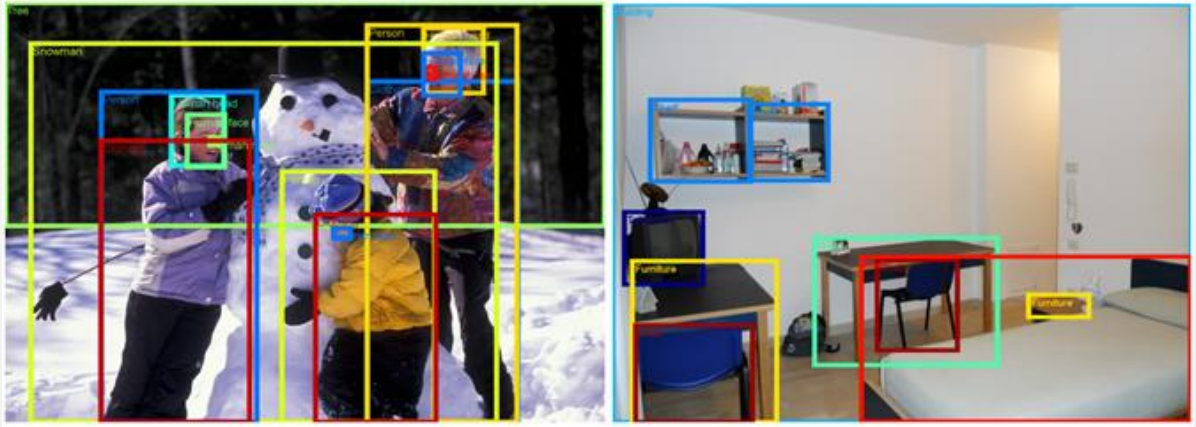
- Pre-trained models on ImageNet-S, including popular convolutional neural networks (CNNs) like VGG, ResNet, and Inception, can be fine-tuned specifically for segmentation tasks.
- Fine-tuning pre-trained models on ImageNet-S enables them to adapt to the nuances of segmentation datasets and achieve superior performance..

ImageNet-S Dataset

Open Image Dataset

Created by Google, Open Images is an open-source image database.

- It boasts over 9 million images, primarily featuring complex scenes.
- Notably, more than 2 million images are hand-annotated with bounding boxes.
- This annotation covers approximately 15.4 million bounding boxes across 600 object classes.
- The dataset hosts the "Open Images Challenge," similar to ImageNet and ILSVRC challenges..



Thank You

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority