Mini-Project 2: Semantic Segmentation with Pascal VOC 2007 Dataset

10/28/2025

Email Contact: mengyu_wang@meei.harvard.edu; mohammad_eslami@meei.harvard.edu

1. Objective

In this project, you will perform semantic segmentation a core task in computer vision where the goal is to assign a class label to each pixel in an image. You will work with the Pascal VOC 2007 dataset, a widely used benchmark containing diverse scenes and object categories.

You will develop, train, and evaluate at least two segmentation models, such as U-Net and the Segment Anything Model (SAM), and compare their performance and behavior across various scenarios.

2. Goals

- Train and evaluate U-Net and SAM (https://ai.meta.com/sam2/) for semantic segmentation.
- Explore at least one additional model or modification, such as DeepLabV3+ or a custom decoder head.
- Compare their performance, training time, and generalization ability.
- Prepare a report including: 1) Methods and model architectures. 2) Results (metrics, qualitative visualizations). 3) Observations and ablation studies. 4) Interpretations and lessons learned.

3. Dataset Preparation

Download the Pascal VOC 2007 dataset from https://www.kaggle.com/datasets/zaraks/pascal-voc-2007

The dataset contains 20 object classes (e.g., person, dog, car, bicycle) plus a background class. It is already split into train, validation, and test sets. In this project, we will use the validation set as our test set and ignore the original test set.

Each object and its corresponding segmentation mask are numerically encoded, as illustrated in Figure 1.

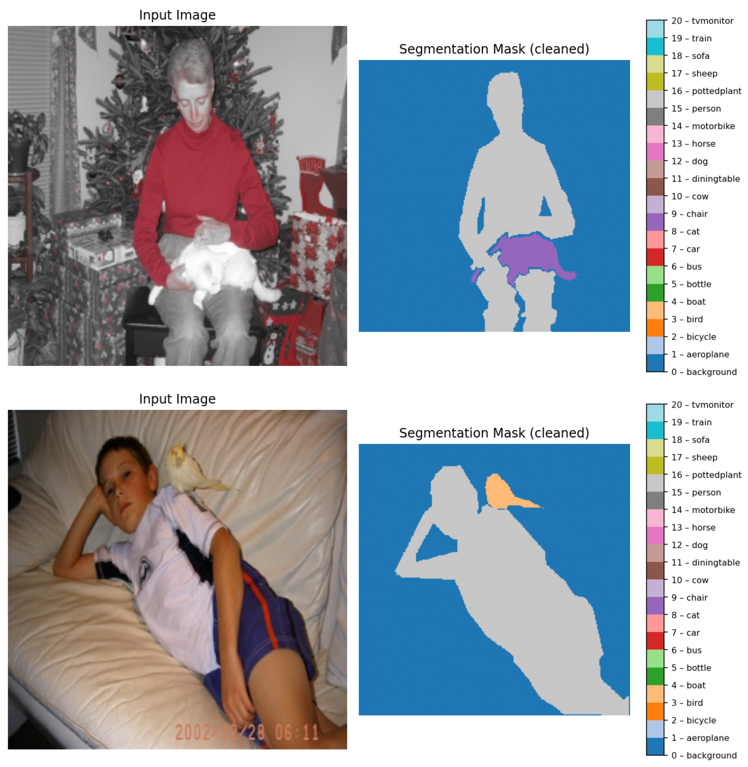

Figure 1: Two random samples from dataset and correspinding labels and masks.

A Python script to load and explore the dataset is provided. (Make sure to update the root path in lines 56 and 65 of the script.) Running this script will load the dataset and display a random image sample along with its class labels and segmentation mask (example outputs in figure 1). Also downloable from https://1drv.ms/u/c/39f018db43a632a0/EbaM0jiYKIVIsj-xZ3Kp454BLxJwhDIw0Lf
e=ghe8KN

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Mon Oct 20 09:17:22 2025
4
5  @author: mohae
6  """
7
8  import torch
9  from torch.utils.data import DataLoader
10 from torchvision import transforms
11 from torchvision.datasets import VOCSegmentation
12 import matplotlib.pyplot as plt
13 import numpy as np
14
15 # -------------------------
16 # 1. Define the 21 classes
17 # -------------------------
18 VOC_CLASSES = [
19     "background", "aeroplane", "bicycle", "bird", "boat", "bottle", "bus",
20     "car", "cat", "chair", "cow", "diningtable", "dog", "horse", "motorbike",
21     "person", "pottedplant", "sheep", "sofa", "train", "tvmonitor"
22 ]
23 NUM_CLASSES = len(VOC_CLASSES)
24 print("Number␣of␣classes:", NUM_CLASSES)
25 print("Classes:", VOC_CLASSES)
26
27
28 # ---------------------------------------
29 # Class mapping dictionary
30 # ---------------------------------------
31 class_mapping = {i: cls for i, cls in enumerate(VOC_CLASSES)}
32
```

```python
print("␣Pascal␣VOC␣2007␣Class␣Mapping:")
for idx, name in class_mapping.items():
    print(f"{idx:2d}␣␣{name}")



# -----------------------------------
# 2. Define transforms for the images
# -----------------------------------
transform_img = transforms.Compose([
    transforms.Resize((256, 256)),           # resize for speed
    transforms.ToTensor(),                   # convert to tensor (C,H,W)
])

transform_target = transforms.Compose([
    transforms.Resize((256, 256)),           # resize mask too
    transforms.PILToTensor()                 # keep as tensor (H,W)
])

# --------------------------------------
# 3. Load the Pascal VOC 2007 Segmentation Dataset
# --------------------------------------
train_dataset = VOCSegmentation(
    root="./VOCtrainval_06-Nov-2007",
    year="2007",
    image_set="train",
    download=False,
    transform=transform_img,
    target_transform=transform_target
)

val_dataset = VOCSegmentation(
    root="./VOCtrainval_06-Nov-2007",
    year="2007",
    image_set="val",
    download=False,
    transform=transform_img,
    target_transform=transform_target
)

print("Train␣samples:", len(train_dataset))
print("Validation␣samples:", len(val_dataset))

# -------------------------
# 4. Create DataLoaders
# -------------------------
train_loader = DataLoader(train_dataset, batch_size=4, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=4, shuffle=False)


# ---------------------------------------
# 5. Inspect a single batch (inputs/outputs)
# ---------------------------------------
images, masks = next(iter(train_loader))
print("Image␣batch␣shape:", images.shape)    # (B, 3, 256, 256)
print("Mask␣batch␣shape:", masks.shape)      # (B, 1, 256, 256)

# ---------------------------------------
# 6. Visualize one image and mask
# ---------------------------------------
def show_sample(img, mask):
    img = img.permute(1, 2, 0).numpy()      # C,H,W -> H,W,C
    mask = mask.squeeze().numpy().copy()    # 1,H,W -> H,W

    # Clean mask: convert all values > 20 (like 255) to 0 for visualization
    mask[mask > 20] = 0

    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(img)
    plt.title("Input␣Image")
    plt.axis("off")

    plt.subplot(1, 2, 2)
    seg_map = plt.imshow(mask, cmap="tab20", vmin=0, vmax=20)
    plt.title("Segmentation␣Mask␣(cleaned)")
    plt.axis("off")

    #  Custom colorbar with class numbers and names
    cbar = plt.colorbar(seg_map, ticks=range(21))
    tick_labels = [f"{i}␣␣{VOC_CLASSES[i]}" for i in range(21)]
    cbar.ax.set_yticklabels(tick_labels)
    cbar.ax.tick_params(labelsize=8)  # smaller font if needed

    plt.tight_layout()
    plt.show()

```

```python
# Show first sample
show_sample(images[0], masks[0])
print("Classes in this mask:", np.unique(masks[0].numpy()))
```

4. Evaluation Metrics

Each model must be evaluated using standard segmentation metrics:

- Mean Dice Coefficient and Intersection-over-Union (mIoU): overall pixel-wise agreement.
- 95th percentile Hausdorff distance (HD95)
- Pixel Accuracy: fraction of correctly labeled pixels.
- Per-class IoU and Accuracy: highlights class-wise strengths and weaknesses.
- Confusion Matrix (optional): pixel-level confusion between classes.

Additionally, visualize:

- Qualitative segmentation maps on a few test images. Mosaic style.
- Side-by-side comparisons between ground truth and predictions. Top three best results, and top three worst results. Consider human class.

5. Ablation Studies

To deepen understanding, run at least two ablation experiments:

- Model size: compare U-Net with different encoder backbones (e.g., ResNet-18 vs ResNet-50).
- Data augmentation: train with vs. without augmentation.
- Loss functions: compare cross-entropy vs. Dice loss.
- Pre-training: train from scratch vs. fine-tuned backbone.
- feel free to suggest other ablations

6. Deliverables

A project report with supplemental code notebooks and scripts. Report completeness, quality, and model performance will be the three grading criteria.

7. Timeline

Submission Deadline: November 18, 2025