Weakly-supervised Semantic Segmentation on Oxford-IIIT Pet with Class-Agnostic Refinement

1 Overview

This framework implements a modular pipeline for weakly-supervised semantic segmentation using the Oxford-IIIT Pet dataset. It supports:

- Training classifiers with various initialization types and CAM methods.
- Generating Class Activation Maps (CAMs)
- Converting CAMs to pseudo segmentation masks
- Optional Class-Agnostic Refinement process (CCAM)
- Training segmentation models using weak or full supervision
- Evaluation and result aggregation

All components are accessible via command-line interfaces in `main.py`.

---

2 Installation

2.1 Additional pip package use declaration

Only one additional package is used beyond the `comp0197-cw1-pt` environment:

- `tqdm`: for progress visualization

2.2 Setup

2.2.1 Create the Environment

Our code is compatible with CPU version of Pytorch, but GPU is **strongly recommend for efficiency**. Change `--index-url` option to `--index-url https://download.pytorch.org/whl/cu124` if you have GPUs.

```
conda create -n comp0197-cw1-pt python=3.12 pip && conda activate comp0197-cw1-pt && pip install torch==2.5.0 torchvision --index-url https://download.pytorch.org/whl/cpu

pip install tqdm
```

2.2.2 Prepare Classifier Checkpoints

We have prepared pretrained classifier checkpoints on Oxford-IIIT Pets:

```
outputs/imagenet_classifier.pth
outputs/mocov2_classifier.pth
outputs/random_classifier.pth
```

These files are available on Google Drive:

https://drive.google.com/drive/folders/1t8Ic0gyOdMZzmWnXhYGwAx1RseX_HMi6?usp=sharing

Especially when you want to try `mocov2` or `random` initialization, please **copy and rename** the corresponding classifier checkpoints and save it **exactly as** `outputs/classifier.pth` to avoid the long training process.

2.2.3 Download Dataset

This framework is configured to work with the **Oxford-IIIT Pet Dataset**.

To download the dataset, run:

```
python main.py download
```

---

3 Running Experiments

Note: This framework uses `ResNet-50` as the default and only backbone for all classification models.

Full Supervision: For full supervision mode (`--supervision full`), the segmentation model is trained using ground-truth pixel masks without relying on CAM-generated pseudo-labels.

All commands below are available via `main.py`. You can run individual steps or the entire pipeline.

3.1 ⭐ Run Full Pipeline (Recommended)

- **Weakly-Supervised Example (GradCAM):**

```
python main.py run_all \
    --init imagenet \
    --cam gradcam \
    --supervision weak
```

- **Fully-Supervised Example (Ground Truth Masks):**

```
python main.py run_all \
    --init imagenet \
    --supervision full
```

3.2 Step 1: Train a Classifier

```
python main.py train_classifier        --init imagenet     --cam gradcam
```

3.3 Step 2: Generate CAM-based Pseudo Masks

```
python main.py generate_masks          --init imagenet     --cam cam       --model_path
{model path to the trained classifier}
```

3.4 Step 3: Train the Segmentation Model

```
python main.py train_segmentation     --supervision weak     --init imagenet     --cam
gradcam     --pseudo_masks_dir {path to pseudo mask}
```

3.5 Step 4: Evaluate Segmentation Performance

```
python main.py evaluate     --supervision weak_gradcam     --init imagenet     --checkpoint
{path to trained segmentation model}
```

4 Customization Options

Below are the customization options that can reproduce the results in our reports.

Option	Values	Description
`--init`	`random`, `mocov2`, `imagenet`	Initialization method
`--cam`	`gradcam`, `cam`, `gradcam+ccam`, `cam+ccam`	CAM methods
`--supervision`	`full`, `weak`	Supervision type

5 Output Structure

```
outputs/
└── experiments/
    └── example_exp/
        ├── masks/
        │   ├── cams/              # CAM heatmaps for each image
        │   └── masks/             # Pseudo segmentation masks generated from CAMs
        ├── best_model.pth         # Trained classifier
        ├── segmentation_best.pth  # Best segmentation model weights
        └── experiment.log         # Full log of training and evaluation
```

6 Project Structure

```
├── main.py
├── train.py
├── generate_masks.py
```

```
├── evaluate.py
├── data.py
│
├── handlers/
│   ├── classifier.py
│   ├── segmentation.py
│   ├── masks.py
│   └── evaluate.py
│
├── models/
│   ├── classifier.py
│   ├── cam.py
│   ├── train_ccam.py
│   └── pspnet.py
│
├── utils/
│   ├── download.py
│   ├── metrics.py
│   ├── visualization.py
│   ├── load_config.py
│   └── logging.py
│
├── config.json
├── requirements.txt
└── README.md
```