Instructions for Weakly Supervised Segmentation with CAM

This document provides detailed steps to run the code and reproduce all reported results.

Additional Packages

Within the comp0197-cw1-pt environment, install the additional required libraries:

```
pip install matplotlib opencv-python pytorch-grad-cam
```

Alternatively, you can use the provided requirements.txt:

```
pip install -r requirements.txt
```

Project Steps

1. Pre-training Classification Models Experiments for Open-Ended Questions
    - Experiment 1: Effect of Irrelevant Samples
    - Experiment 2: Effect of Multi-task Training
2. CAM Evaluation
3. Weakly-Supervised training using CAMs as only supervision
4. Self-Supervised training on the weakly supervised stage (Alternative Open-Ended Question) Experiments:
    - Experiment 3: Effect of Self-training with Augmentations
5. Evaluation of Segmentation Models

Running the pipeline

The entire pipeline can be run as

```
python main.py
```

The following flags in `main.py` control which parts of the pipeline will actually run:

- `TEST_MODELS_BEFORE_TRAINING` -> Whether to test if the models can actually be initiated (recommended False)
- `PRETRAIN_MODELS` -> Whether to run the pre-training process, including all experiments
- `EVALUATE_CAMS` -> Whether to evaluate the CAMs generated from the pre-training process
- `GENERATE_CAM_DATASET` -> Whether to generate a fresh dataset from the best CAM in the previous step

- `TRAIN_FULLY_SUPERVISED` -> Whether to train a new fully supervised baseline model
- `TRAIN_WEAKLY_SUPERVISED` -> Whether to train a new weakly supervised model, with the best CAMs as supervision
- `TRAIN_SELF_TRAINING` -> Whether to run the self-training process, including all experiments
- `EVALUATE_MODELS` -> Whether to run the script that evaluates the baseline model

Additionally, you can set the variables for the number of workers, batch_size, and whether to use persistent workers and pin memory according to the device you have available in the section market as "Set device" (lines 50-66 of `main.py`), but note that some task use different set ups due to their memory intensity, those can be updated in the respective function call.

Additionally, you can use `config.py` to control the actual experiments to run; the key variables are:

- `RUNS_CONFIG` -> controls the experiments of the pre-training models; which we call runs
- `MODEL_NAMES` -> controls the models that will be pre-trained (the format is backbone_[list_of_head_targets])
- `SELF_LEARNING_EXPERIMENTS_CONFIG` -> controls the experiments to run on the self-learning stage
- `DATASET_SIZE` -> the number of items to include in the dataset (total fo train, test and val); setting as None will use the entire dataset
- `DEFAULT_IMAGE_SIZE` -> the size of images to use across the entire pipeline.

If you want to do a test run using the entire pipeline, we recommend setting the `DATASET_SIZE` to 10 and reducing the number of epochs (`PRETRAIN_NUM_EPOCHS` in `main.py`). If you will use the entire dataset we recommend running one section at a time (so, a single flag set as True at a time).

Directory Structure

Key files:

Modules:

- `cam_generation/`: Contains all code related to the generation and evaluation of Class Activation Maps
- `datasets/`: Contains all code related to initiating the Dataset and Dataloader classes for the project
- `models/`: Contains all the models used in training (be it pre-training, weakly-supervised, etc.)
- `training/`: Contains all code related to the actual training process, including the scripts used for each stage of training, modules for evaluation functions, etc.

Storage:

- `cam_datasets/`: Storage of generated datasets with CAMs as labels (recommended to only keep one at a time here)
- `checkpoints/`: Saved model checkpoints (we will not be uploading any due to file sizes)
- `data/`: Storage of base datasets (e.g., Oxford Pet III)
- `kaggle/`: Kaggle configuration to download datasets
- `logs/`: Training logs (we will not be uploading any due to file sizes)
- `visualizations/`: Generated visualization outputs