Project Code Setup Guide

April 12, 2024

1 Introduction

This document provides all the necessary instructions and steps required to run the code submitted by our team. Please follow the steps below to ensure that you can replicate our experiment results.

2 Environment Setup

2.1 Creating a conda environment

Open a terminal or command prompt and run the following command to create a new conda environment:

```
conda create -n comp0090-cw1-pt -c pytorch python=3.12 pytorch=2.2
    torchvision=0.17
```

2.2 Installing necessary packages

We provide a requirements.txt: cv2, imgaug, pillow

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

3 Project Structure

- **Python Scripts**

  `data_utils.py` Handles data loading and preprocessing tasks, ensuring the data is correctly formatted and ready for both training and test phases.

  `data_augmentation.py` Contains code for augmenting the pre-training datasets.

  `models.py` Defines the masked auto encoder network models used for segmentation.

`losses.py` Provides definitions for the loss functions used to train the models.

`metrics.py` Defines metrics used to evaluate the performance of the models during and after training.

`utils.py` Includes utility functions that are used across the project, such as visualization tools to analyze model performance and training results.

- **Directories**

`datasets/` This directory stores the origin datasets and augmented data prepared for use in training and testing the models.

`experiments/` Contains separate code files for each experiment conducted, including:

  - `compare_pretrained_with_baseline`
  - `compare_pretrained_model_finetuning_sizes`
  - `compare_data_similarity_for_segmentation`

  Each script is dedicated to a specific analysis or comparison.

`Images/` Holds visualization images that depict the results of experiments.

4 Running Code & Reproducing the Experiments

Here are the steps to run our project code, the experiment code files are stored in 'experiments/', outlines the steps to replicate the experiments . Ensure you have set up the environment as described in the previous section.

4.1 compare_pretrained_with_baseline.py

The purpose of this experiment is to compare the framework with a baseline model trained on the same finetuning data, using fully supervised methods. Only a pretrained dataset needs to be prepared, located at '/datasets/data' to directly run the code.

4.2 compare_pretrained_model_finetuning_sizes.py

The purpose of this experiment is to compare the benefit of the pretrained segmentation model, using different finetuning dataset sizes. Only a pretrained dataset needs to be prepared, located at '/datasets/data' to directly run the code.

4.3 compare_data_similarity_for_segmentation.py

The purpose of this experiment is to explore how similar the pretraining and finetuning/test data need to be for a better segmentation model. A pretrained dataset related to pets is prepared, located at '/datasets/data', and another unrelated pretrained dataset, located at '/datasets/102flowers', needs to be prepared to run the code.