Leiden University
Computer Vision, Autumn 2025
Instructor: Hazel Doughty, Lu Cao

Assignment 2: Scene Recognition with CNNs

1 Goal

The goal of this assignment is to build and train a scene classifier for images using a CNN. This is an individual assignment.

2 Data

The dataset we'll use is the MIT Mini Places dataset, consisting of images from one of 100 scene categories. It's a subset of the much larger MIT Places2 dataset. The training set of the Mini Places dataset has 100,000 images; the validation and test sets have 10,000 images each. We will provide you with the images for all sets but only labels for the train and validation sets, the test labels will not be provided.

You can download the data using the command:

```
wget https://web.cs.ucla.edu/~smo3/data.tar.gz
```

make sure to move extract the images and move them to `data/images` to make sure the starter code can load them.

3 Starter Code

We have provided starter code which can be run using

```
python scene_classification.py
```

Currently, the code just loads the data. To make the code work you need to define an optimizer, a criterion and the model. Once you've done this you can run your code to train the model and it will save the model weights to `model.pth`. To obtain predictions for the test images you can run the code as:

```
python scene_classification.py --test
```

This will load the model weights and make predictions for the test images.

4 Rules

You may **not** use outside data in your submission. This includes using pre-trained models, e.g. models first trained on Imagenet then fine-tuned on Mini Places.

You must use at **least 2 convolutional** layers in your network. However, your network does not have to be exclusively convolutional layers; CNNs typically use pooling and fully connected layers as well.

Your solution should be able to run (both testing and training) on the machines in the computer rooms DM.0.07, DM.0.13, DM.0.17, DM.0.21, i.e. they can use up to **8GB of GPU memory**.

Besides these rules you are free to do whatever you want.

5 Submission

Submit your new `scene_classification.py` file to BrightSpace along with any other necessary files .py files that you have created, your model checkpoint and the predictions.csv outputted by running your code in test mode. The automarker will compare your predictions to the ground-truth predictions and also run your code with the commands:

```
python scene_classification.py
python scene_classification.py --test
```

to ensure your training and testing runs as expected.

6 Grading

Your grade depends on the accuracy your model achieves on the test data. 25% accuracy will obtain a 6, 40% will obtain 8. We will do linear interpolation to obtain the grades for other results.

7 Running your code on the lab machines

It may be useful to run your code on the lab machines in DM.0.09, etc. This is relatively easy to set up. First create a conda environment with:

```
conda create --name pytorch python=3.9
```

activate it with

```
conda activate pytorch
```

and install pytorch with

```
conda install pytorch torchvision torchaudio pytorch-cuda=12.4 -c pytorch -c nvidia
```