Assignment 13

Applied Machine Learning

In this assignment, we will work on image classification using PyTorch.

- Download the intel image dataset from Kaggle.
- We will use the OpenCV image feature extraction library.
  (`conda install -c conda-forge opencv`)

1. [20 pts] Download the dataset, unzip and explore the file folders. Load the image dataset with training and testing grouped. (Note, `cv2` reads and saves in BGR channel order)

```
import cv2

IMGSIZE = (128, 128)
CNAMES = ['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']

X_tr, y_tr, X_ts, y_ts = [], [], [], []
for label in _labels:
    path = _path + '/seg_train/seg_train/' + label
    for f in sorted([_ for _ in os.listdir(path) if _.lower().endswith('.jpg')]):
        X_tr += [cv2.resize(cv2.imread(os.path.join(path,f)), IMGSIZE)]
        y_tr += [CNAMES.index(label)]
```

   Display a few images. How many color channels are there?

2. [20 pts] Convert the imageset to `numpy` array, such as the array size:
   `(14034, 128, 128, 3)`
   Scale the imageset to [0-1].

3. [50 pts] Create a neural network to train and report its performance on the testing portion of the dataset. 60% reclassification and 55% testing performance should be achievable without any hyperparameter tuning. (Hint: My model with (50,50,50) layers, which is similar to the model in module notebook, took around 5 minutes to train 200 epochs.)

   Reminder, you have to convert the 3D image (including the color channel) to a linear vector in case you use fully connected layers at the input, i.e.
   `_X = torch.flatten(_X, start_dim=1)`

   (Hint, for debugging use `print` statements or PyCharm to display the tensor shapes)

4. [10 pts] Research and find out how to improve the neural network model to solve this problem. Outline an approach.