Assignment 8

Image Classification using CNN and Resnet

Computing Lab II, Spring 2022

The objective of this assignment is to train a classifier to identify land use and land cover. You will implement a Convolutional Neural Network (CNN) and fine-tune Resnet model for this image classification task. Use **PyTorch** and **kaggle** for this assignment.

Dataset: EuroSAT Dataset (RGB)[1]
EuroSAT Dataset consists of remotely sensed satellite images with 10 different categories of 27,000 color images of size 64×64 pixels. Use the RGB version of the dataset.

Class Labels: Each image is assigned to one of the following labels:

- 0 AnnualCrop
- 1 Forest
- 2 HerbaceousVegetation
- 3 Highway
- 4 Industrial
- 5 Pasture
- 6 PermanentCrop
- 7 Residential
- 8 River
- 9 SeaLake

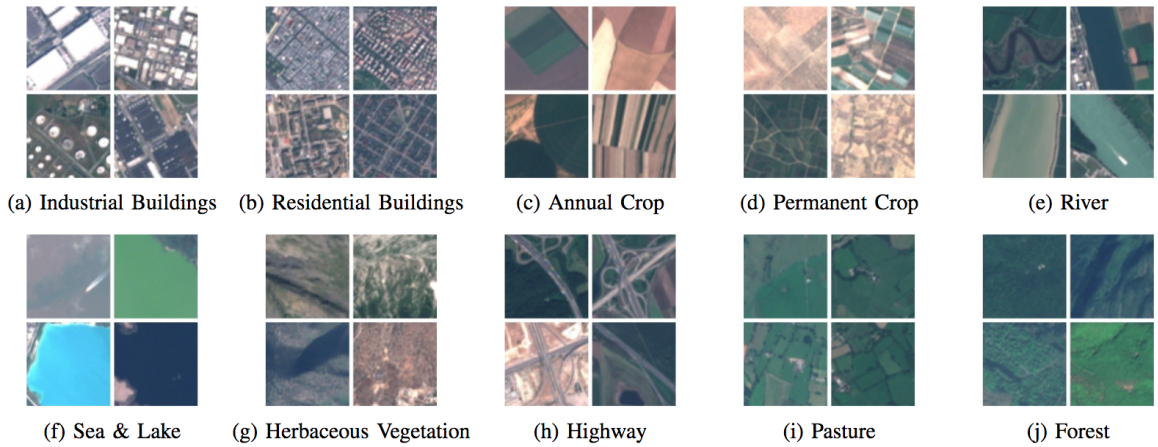


Fig: This overview shows sample images of all 10 classes in the EuroSAT dataset. Each class contains 2,000 to 3,000 images.

The train, validation, and test datasets are here. The ratio of the splits is ~ 70:10:20. After unzipping the euroSAT_tvt_pkl.zip file, you will get the pickled file euroSAT_train_val_test.pkl. A sample script to load the dataset is shown below:

```
import pickle
with open("euroSAT_train_val_test.pkl", 'rb') as f:
    train_set, val_set, test_set = pickle.load(f)
print (len(train_set), len(val_set), len(test_set)) # 18792 2808 5400
print (train_set[0][0].shape, train_set[0][1]) # (3, 64, 64) 7
```

You should record both the training and validation errors at the end of each epoch (do not train on the validation set, just record the error). Use the **early stopping** technique to decide when to finish training. A simple algorithm for this would be as follows:

1. Train on training data.
2. Find validation set performance at regular intervals.
3. If validation set performance doesn't improve over k epochs (called patience), stop training. This is called **early stopping.**

After the model is trained with early stopping**,** load best model weights based on validation set and find performance on the test set.

Task 1 : CNN

You will create a 2D convolutional neural network using `torch.nn.conv2d` as a basic component. Train it end to end.

CNN specifics:

- 1st convolutional layer: 64 out channels, kernel size 3x3, stride 2, padding 1, followed by relu activation, and max pooling (with 2x2 subsampling)
- 2nd convolutional layer: 96 out channels, kernel size 3x3, padding 1, followed by relu activation, and max pooling (with 2x2 subsampling)
- 3rd convolutional layer: 192 out channels, kernel size 3x3, padding 1, followed by relu activation and max pooling (with 2x2 subsampling).
- Flatten the above layer and feed it to a linear layer with 1024 hidden units followed by relu activation
- Add dropout with retention probability of 0.5
- Again feed it to a linear layer with 512 hidden units followed by relu activation.
- Add an output dense layer with 10 nodes
- Loss function: cross entropy loss
- Optimizer: Adam

Deliverables

1. Save the weights corresponding to the **minimum validation loss** in a folder 'logs_cnn/'
2. Load the saved model and report the accuracy on the validation and test set.
3. Create plots of loss and accuracy on the training and validation sets.
4. Plot the confusion matrix heatmap of the test set.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html

Submit a brief report on what inferences you could draw from the experiments (in pdf only). The report should contain information on model hyperparameters (i.e. batch size, learning rate, epoch, patience etc.) and the training, validation and test accuracies.

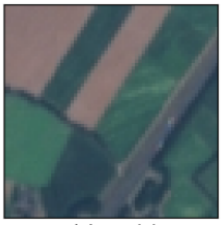
Task 2

You will finetune the Resnet (Residual Networks)[2] model. Resnet is trained on the ImageNet dataset and you will use it to finetune on the EuroSAT dataset. In **finetuning**, we take a model trained on Dataset D1, and then train it on a new Dataset D2. We can update all parameters of the model or keep some of the layers frozen. You will fine tune the Resnet18 model with the training data of task1. For validation, use the validation data of task1.

- Load the pretrained **Resnet18** model from `torchvision.models`.Keep the parameter `pretrained=True`
- Add an output dense layer with 10 nodes
- Loss function: cross entropy loss
- Optimizer: Adam
- Use a low learning rate and adjust it with grid search

Deliverables

1. Save the weights corresponding to the **minimum validation loss** in a folder 'logs_resnet/'
2. Load the saved model and report the accuracy on the validation and test set
3. Create plots of loss and accuracy on the training and validation sets.
4. Show the recall for each class.
5. Plot four examples the model got wrong and was most confident about**.** Also add the true class and the predicted class along with the image. One such example is shown below.



True : Highway Predicted: Forest

6. Retrain the model with the same hyper parameters, but keep `pretrained=False.` Report the accuracy. Is there any difference when `pretrained=True` and `pretrained=False`?

Submit brief reports on what inferences you could draw from the experiments (in pdfs only). The reports should contain information on model hyperparameters (i.e. batch size, learning rate, epoch, patience etc.) and the training, validation and test accuracies.

Constraints and Considerations for Both the Tasks:

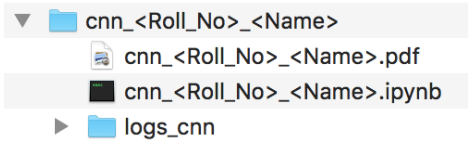
1. In each experiment, set the seed value as the last two digits of your roll number for reproducing your result.
   https://pytorch.org/docs/stable/notes/randomness.html

2. You must train properly with a proper stopping criterion. All decisions you take for model parameters must be logical.

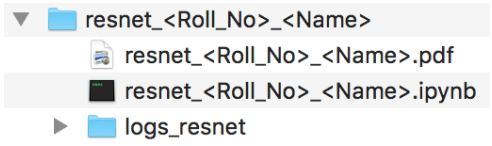
3. Do NOT hardcode any paths. Include all training and testing codes in a jupyter notebook. Make sure that code for all the deliverables are also included in the notebook.

4. You will use **kaggle** to submit the prediction files.
   a. Join the following competitions in kaggle
      i. Task1
      https://www.kaggle.com/account/login?returnUrl=%2Ft%2F8c49c19347a44503afd4c51687cf6af3
      ii. Task2_part1 (`pretrained=True`)
      https://www.kaggle.com/competitions/ml-assignment-cl-ii-lab-task2-part1/overview
      iii. Task2_part2 (`pretrained=False`)
      https://www.kaggle.com/competitions/ml-assignment-cl-ii-lab-task2-part2/overview
   b. Change your team name to <Roll_No>_<Name>
   c. Submit the prediction files in the required format for leaderboard entries. (You can submit a maximum of 20 times in a day). ***Your assignment will not be evaluated if it does not appear in the leaderboard.***

5. In the **Moodle**, submit two **ZIP files** cnn_<Roll_No>_<Name>.zip and resnet_<Roll_No>_<Name>.zip for task1 and task2 respectively. Note that each zip file should contain the following files/folders:



cnn_<Roll_No>_<Name>.zip



resnet_<Roll_No>_<Name>.zip

   a. your assignment codes, i.e., **jupyter notebook**s
   b. **model weights** (i.e. the 'logs_cnn/' and 'logs_resnet' folders)
   c. **pdf file** explaining your model hyperparameters and results/inferences. Please note that all explanations should be short (1-3 sentences).
      i. *cnn_<Roll_No>_<Name>.pdf* should contains the deliverables 2 to 4 for task1
      ii. *resnet_<Roll_No>_<Name>.pdf* should contains the deliverables 2 to 6 for task2

6. If the weights are exceeding the upload constraints imposed by Moodle (may not happen), then upload the weights in some public site like github and download them first, by writing appropriate code for it.

7. Small differences in your method's accuracy would not be penalized (larger may be). Your experimentation technique should be sound (like, do not test on training data or train on validation data).

[1] Helber, Patrick, et al. "Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification." IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 12.7 (2019): 2217-2226.

[2] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.