1 Introduction

This assignment is divided in two parts. In the first part of this assignment it is expected that you build a CNN model from scratch and evaluate it in a public dataset. In the second part, it is expected that you train and test two well-known networks and use their output for classification of images using a late fusion strategy. This assignment is part of your continuous evaluation. You should provide a report and a notebook (.ipynb) which includes your development and analysis for the requested tasks. Structure your report in accordance with the assignment parts.

2 Datasets

In this assignment you will be using the CIFAR-10[1] dataset and the EuroSAT[2] dataset. The CIFAR-10 dataset is used for image classification and is composed of a diverse number of objects, animals and other entries, totaling 10 distinct classes (plane, car, bird, cat, deer, dog, frog, horse, ship and truck). For each class there are 6000 colored (RGB) images ($3 \times 32 \times 32$, i.e. 3-channel color images of 32×32 pixels), where 5000 images are used for training and 1000 images are used only for testing. Similarly, the EuroSAT dataset is used for image classification, in particular, land use and land cover classification using Sentinel-2 satellite images, with a total of 10 distinct classes (AnnualCrop, Forest, HerbaceousVegetation, Highway, Industrial, Pasture, PermanentCrop, Residential, River and SeaLake). The dataset contains 27000 colored (RGB) images ($3 \times 64 \times 64$, i.e. 3-channel color images of 64×64 pixels). Training and testing splits are not included in the dataset.

2.1 Downloading the datasets

To download the CIFAR-10 dataset you can use the following code:

```python
from torchvision.datasets import CIFAR10
import torchvision.transforms as tt
import numpy as np

transform = tt.ToTensor()

training_data = CIFAR10(download=True,root="./data",transform=transform)
testing_data = CIFAR10(root="./data",train=False,transform=transform)
```

To download the EuroSAT dataset you can use the following code:

```python
from torchvision.datasets import EuroSAT
import torchvision.transforms as tt
import torch
import numpy as np

transform = tt.ToTensor()

#To solve local certificate errors
#import ssl
#ssl._create_default_https_context = ssl._create_unverified_context
dataset = EuroSAT(download=True,root="./data",transform=transform)
#define TrainingSIZE and TestingSIZE considering the desired split
#(e.g., TrainingSIZE=int(len(dataset)*0.5)).
training_data, testing_data = torch.utils.data.random_split(dataset, [TrainingSIZE, TestingSIZE])
```

3 Network

Figure 1 shows a simple configuration of a CNN architecture. It is composed of three convolutional layers followed by three fully-connected layers. Using the following hyper-parameters (lr = 0.0001; epochs = 25), this network achieved 64% of mean-Class Accuracy on the CIFAR-10 dataset.

[1] https://www.cs.toronto.edu/~kriz/cifar.html
[2] https://github.com/phelber/eurosat

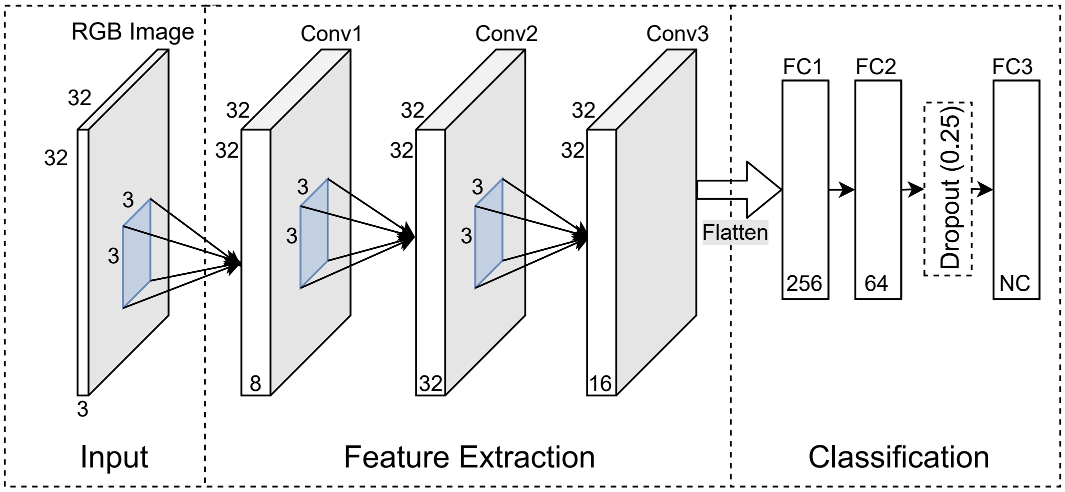

Figure 1: Convolutional Neural Network baseline architecture. NC represents the training number of classes.

3.1 Evaluation Metrics

Evaluating a CNN model is a crucial part in the design of an effective learning model. In classification tasks, mean-Class Accuracy (mCA) is the most used evaluation metric, and is represented as follows:

$$mCA = \frac{1}{N}\sum_{i=1}^{N}\frac{Correct_i}{Total_i}$$

where $Correct_i$ is the number of correctly predicted samples of class i, $Total_i$ is the total number of samples of class i, and N is the total number of classes.

Other important evaluation metrics include the F1-Score, Precision and Recall which where already addressed in a previous class. For a better analysis of the trained model, the confusion matrix should also be used.

4 Tasks

Part I

- Modify the baseline network (Fig. 1) to reach an output mCA of at least **75%** on the CIFAR-10 dataset. To accomplish this you can:
  - add/remove convolutional layers;
  - add/remove fully-connected layers,
  - add/remove dropout or add other regularization methods;
  - add max-pooling layers;
  - modify the activation functions;
  - change the loss function and/or change the optimization method;
  - add batch normalization layers;
  - change any hyper-parameter;
  - you can use an adaptive learning rate;
  - your network cannot have more than **five** convolutional layers;
  - you can also add data augmentation techniques on your dataset.

- Implement the above-mentioned evaluation metrics to evaluate your network.
- Gather the metrics and loss curves for four different hyper-parameter combinations (learning rate, batch size, etc)

PART II

- Implement both the AlexNet and MobileNetV2 networks (you can import the network from PyTorch);
- Define a training/test split for the EuroSAT dataset (e.g., 80/20);
- Train both networks from scratch (both networks should be trained using the same training hyper-parameters);
- Train both networks using, as a starting point, the publicly available pre-trained model of the network you are using, in this case, pre-trained with the ImageNet dataset (for this you have two options: you can download the model and copy the weights to your network, or you can also use the "torchvision.models" library available in PyTorch);
- Using the two trained models from the previous transfer learning step, implement a late fusion technique. Note that a late fusion technique does not require a training process. It uses the prediction vectors of each network (networks' output) and makes a decision from their combination (e.g., summation, averaging, multiplication, Bayesian combination). **You can decide on which late fusion technique you want to use**. For a starting point on late fusion techniques, take a look at the paper available in `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9190246`;
- Implement the aforementioned evaluation metrics to evaluate your networks and the chosen late fusion;
- Gather the metrics and loss curves for all training conditions.

Tip: To avoid Colab virtual machine connection issues, save your models after the training phase is completed. To save your CNN models, you can save them on your Google drive or download them to your machine.

5 Deliverables

Each group (two students) must deliver the following material:

- A detailed report (**cannot exceed 10 pages**) with:
  - **PART I**
    - diagrams with the implemented CNN architecture accompanied with descriptions and the thought process when designing the network. You should write any additional implementation details as well as any diagrams you see fit to highlight the different stages of your implementation (e.g., data loading, training, validation, evaluation). Hyper-parameter values should also be included when relevant;
    - a table with mCA, F1-Score, Precision, Recall, a plot with the achieved loss curves, and a confusion matrix for the best achieved result;
    - a comparison and a detailed analysis of the results obtained using four different hyper-parameters (consider possible overfitting or underfitting scenarios);
  - **PART II**
    - describe the training protocol for both networks using pre-trained weights and when trained from scratch;
    - evaluation metrics, loss curves, and confusion matrices for AlexNet (pre-trained and scratch) and MobileNetV2 (pre-trained and scratch); Due to the computational burden of both methods and Colab limitations you can select a suitable number of training epochs (should be the same for all methods).

    - present the late fusion approach followed and the obtained results (can be included in the results reported in the previous item);
    - detailed comparative analysis of the results obtained;

- Google Colab notebook (ipynb)