1 Introduction

This assignment is divided into 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 image classification, using a middle-fusion strategy.

This assignment should be done in groups of 2 students max (mandatory), and it is part of your continuous evaluation. You should provide a report, with no more than 10 pages, and a notebook (.ipynb/.py) 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 STL-10[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 STL-10 dataset is used for image classification and was designed to evaluate unsupervised and semi-supervised learning methods. It consists of 10 distinct classes (airplane, bird, car, cat, deer, dog, horse, monkey, ship and truck), similar to those in CIFAR-10. The dataset contains RGB images of higher resolution, each sized $3 \times 96 \times 96$ (i.e. 3-channel color images of 96×96 pixels). Unlike CIFAR-10, STL-10 provides a limited number of labeled images (a total of 5000 for training and 8000 for testing) along with an additional 100,000 unlabeled images intended for unsupervised feature learning. Considering the smaller dataset size (unlabeled samples are not relevant for this assignment), STL-10 models trained from scratch should present poor performances. Validation splits are not included in the datasets.

2.1 Downloading the datasets

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

```
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)
train_set, val_set = torch.utils.data.random_split(training_data, [int(len(training_data)*0.8), int(
    len(training_data)*0.2)])
```

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

```
from torchvision.datasets import STL10
import torchvision.transforms as tt

transform = tt.ToTensor()

training_data = STL10(root='./data',  split='train', download=True,   transform=transform)
testing_data = STL10(root='./data', split='test', download=True,   transform=transform)
train_set, val_set = torch.utils.data.random_split(training_data, [int(len(training_data)*0.8), int(
    len(training_data)*0.2)])
```

[1] https://www.cs.toronto.edu/~kriz/cifar.html
[2] https://cs.stanford.edu/~acoates/stl10/

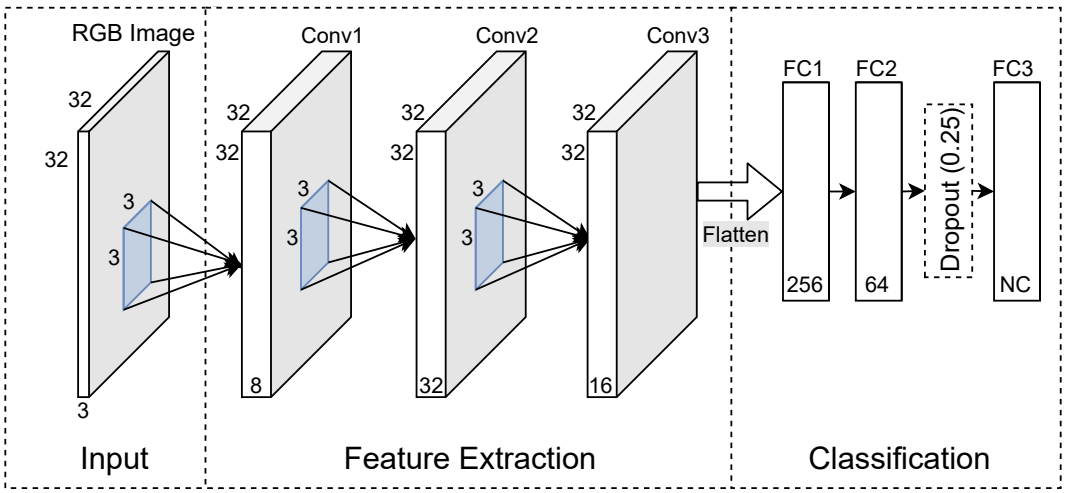

Figure 1: Convolutional Neural Network baseline architecture. NC represents the training number of classes.

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 hyperparameters (lr = 0.0001; epochs = 25), this network achieved 64% of mean-Class Accuracy on the CIFAR-10 dataset.

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 were 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 **82%** 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 hyperparameter;
  - you can use an adaptive learning rate;
  - your network cannot have more than **six** convolutional layers;
  - add data augmentation to your dataset.

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

PART II

- Implement in your code one ResNet model (18, 34, 50, 101 or 152) and one MobileNet V3 (small or large) model (you can import the networks from PyTorch - torchvision.models[3] or timm[4]). Take into account your computational resources in this step, as well as the implications of this choice e.g., choosing one complex architecture in detriment of a simpler architecture, for this specific dataset;
- Define a training/validation split for the STL-10 dataset (e.g., 80/20);
- **Train** both networks you selected from scratch (both networks should be trained using the same training hyperparameters);
- **Train** both networks using, as a starting point, the publicly available pretrained model of the network you are using, in most cases it will be an ImageNet-trained model (check torchvision or timm pretrained weights);
- Using the two best trained models (one ResNet and one MobileNetV3) implement a middle fusion technique. You should create a new network that combines the feature maps from the two different models, and create a new classification layer that takes as input the feature maps. The feature maps can be aggregated or concatenated. For each model you should freeze the feature extraction weights (otherwise explain why).
- **IMPORTANT:** Implement the aforementioned evaluation metrics to evaluate your networks and the chosen middle fusion (check sklearn);
- **IMPORTANT:** 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 (max two students) must deliver the following material:

- A detailed report **not exceeding 10 pages** (can follow an IEEE two-column template but that is not mandatory) with:
  - **PART I**
    - Diagrams (e.g., draw.io) with the implemented CNN architecture accompanied by 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). **Hyperparameter values should also be included when relevant;**

[3] https://docs.pytorch.org/vision/main/models.html
[4] https://huggingface.co/docs/timm/index

  - 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 hyperparameters (consider possible overfitting or underfitting scenarios);
- **PART II**
  - Describe the training protocol for both networks using pretrained weights and when trained from scratch;
  - Evaluation metrics, loss curves, and confusion matrices for the ResNet (pretrained and scratch) and MobileNetV3 (pretrained and scratch) models; Due to the possible 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 middle fusion approach followed and the obtained results (can be included in the results reported in the previous item, i.e., a single table with all metrics). Include a diagram to represent the middle fusion approach;
  - Detailed comparative analysis of the results obtained;

- Google Colab notebook (ipynb) or python file (py)