# Deep Learning (85837) - 2024-2025 - Ex 3

Due: 10.2.2025, 11:59pm

## The Mission

In this exercise, you will need to solve a multi-class classification task with different types of Neural Networks using PyTorch. The dataset you will be working on is STL-10.

STL-10 (https://cs.stanford.edu/ acoates/stl10/) is a dataset of images, consisting of 500 labeled training images and 800 labeled test images per class. Images are 96x96 pixels from the following 10 classes: Airplane, Bird, Car, Cat, Deer, Dog, Horse, Monkey, Ship, Truck.

As we saw in class, PyTorch has built-in tools for reading datasets and manipulating them. You will need to use them in this exercise. Specifically, the STL-10 dataset is available through the PyTorch Datasets class (https://pytorch.org/docs/stable/torchvision/datasets.html). In this exercise, we will use only the 'train' and 'test' splits. Note that you will need to split on your own the training set into a train part and a validation part.

## Part 1 - Visualize the Data (10 Pts)

First, you will need to visualize the data using the Matplotlib package. Plot 4 different examples from each class in a grid of $10 \times 4$ (i.e., 10 rows, each contains 4 different examples from the same class). Label each row with the class name. Add the figure to a report that should be submitted along with your code.

## Part 2 - Classification with Various Networks (90 Pts)

Here, you will experiment with different types of networks. To reduce the computational complexity we will be working with images of size $3 \times 64 \times 64$. During training apply random cropping for images to size 64, and for test images center cropping to size 64. These operations can be done easily using the PyTorch Transforms class. So, to clarify, the input to the network (both during training and testing) should be an image of size $3 \times 64 \times 64$. Also, in all experiments, during training, you need to apply data augmentation techniques (such as random horizontal flipping). You may choose whichever augmentations you want, but you have to apply at least two augmentations. Describe in the report which augmentations were selected and add one image that was transformed by each augmentation.

Implement the following networks:

1. Logistic regression over flattened version of the images
2. Fully-connected NN with at least 3 hidden layers over flattened version of the images followed by a classification layer. Apply batch normalization and dropout to all hidden layers.
3. CNN with at least two convolution layers and two pooling layers followed by two fully connected layers and a classification layer. Apply batch normalization to the convolution layers and dropout to the fully connected layers.
4. A fixed pre-trained MobileNetV2 as feature extractor followed by two fully connected layers and an additional classification layer. To clarify, you need to learn only the parameters of the task head.
5. A learned pre-trained MobileNetV2 as feature extractor followed by two fully connected layers and an additional classification layer. Here, you need to learn the parameters of the MobileNetV2 as well.

For this exercise, you are advised to use google-colab or any other GPU hardware at your disposal. Note that to move tensors from the server memory to the GPU you will need to use either the .cuda() command or .to(#device) command. Further instructions can be found in PyTorch documentation site.

Test different configurations of hyperparameters: layer size, optimizer, batch size, learning rate, regularization coefficient, etc. Write your findings in a report (several lines describing your main insights should suffice). Add to the report a plot of the train and validation loss and accuracy curves as a function of the number of epochs of your best model for each of the networks described above and the test accuracy for each of these models. Also, for networks 2 and 3 report the number of learnable parameters (give a detailed description of the calculation, just a number will not suffice).

## Final Remarks

1. No package restrictions in this exercise.
2. You need to submit a report and your code. Add to the report clear instructions on how to run your code.
3. Submission instructions (similar to the ones in previous exercises):
    (a) Code should be submitted in .py or .ipynb file format only.
    (b) Add to the report your name and id.
    (c) Pack your submission files with your favorite file archiver (e.g., .rar, .zip).
    (d) The archive name should be your ID. If the exercise is done in pairs, the name of the file should be in the following format ID1_ID2. Only one member needs to submit the solution.

## Good Luck