Introduction to Computer Vision (ECSE 415)
Assignment 4: Neural Networks

DEADLINE: March, 15
Please submit your assignment solutions electronically via the **myCourses** assignment dropbox.

The submission should include a single Jupyter notebook. More details on the format of the submission can be found below. Submissions that do not follow the format will be penalized 10%.

The assignment will be graded out of a total of **100 points**. There are *50 points* for accurate analysis and description, *40 points* for bug-free and clean code, and *10 points* concerning the appropriate structure in writing your report with citations and references.

Each assignment will be graded according to defined rubrics that will be visible to students. Check out MyCourses, the "Rubrics" option on the navigation bar. You can use **OpenCV**, **sklearn**, **skimage**, **Numpy**, and **Pytorch** (Pytorch is recommended, but TensorFlow is also accepted) library functions for all parts of the assignment except those stated otherwise. Students are expected to write their own code.).**here**).

Assignments received late will be penalized by 10% per day.

Submission Instructions

1. Submit a single Jupyter notebook consisting of the solution of the entire assignment.
2. Comment your code appropriately.
3. Give references for all codes which are not written by you. (Ex. the code is taken from an online source or from tutorials)
4. Do not forget to run **Markdown** ('Text') cells.
5. Do not submit input/output images. Output images should be displayed in the Jupyter notebook itself.
6. Make sure that the submitted code is running without error. Add a **README** file if required.
7. If external libraries were used in your code please specify their name and version in the **README** file.
8. We are expecting you to make a path variable at the beginning of your codebase. This should point to your working local (or google drive) folder.
   Ex. If you are reading an image in the following format:

   ```
   img = cv2.imread ( '/content/drive/MyDrive/Assignment1/images/shapes.png' )
   ```

   Then you should convert it into the following:

   ```
   path = '/content/drive/MyDrive/Assignment1/images/'
   img = cv2.imread(path + 'shapes.png')
   ```

   Your path variable should be defined at the top of your Jupyter notebook. While grading, we are expecting that we just have to change the path variable once and it will allow us to run your solution smoothly. Specify, your path variable in the **README** file.
9. Answers to reasoning questions should be comprehensive but concise.

1 Part 1 - CIFAR-10 Classification using Convolution Neural Network (70 points)

In this section, you are going to train models on the publicly available CIFAR-10 dataset **source**. The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. For more information, you are encouraged to look at their webpage. You are expected to implement a Convolution Neural Network (CNN) to classify the images based on their context. **Free T4 GPU in Colab is highly recommended for executing the code in this section**.

1. Implement a shallow CNN with the layers mentioned below.
    - A Convolution layer with 32 kernels of size 3x3
    - A ReLU activation
    - A Convolution layer with 64 kernels of size 3x3
    - A ReLU activation
    - A maxpool layer with kernels size of 2x2
    - A convolution layer with 64 kernels of size 3x3
    - A ReLU activation
    - A convolution layer with 64 kernels of size 3x3
    - A ReLU activation
    - A flattening layer. (This layer resizes a 3D tensor to a feature vector).
    - A fully connected layer with an output size of 10. (Classes should be predicted as numerical values (like 0-9).

   Note: If you use Pytorch, you can use *print(next(model.parameters()).device)* to check if you're using the GPU for training.

2. Use Pytorch Class *torchvision.datasets.CIFAR10* to load the dataset.
3. Training, validation and test settings are shown below.
    - 50,000 images for training (training set). Divide the 10,000 test set images of CIFAR10 into two subsets by 1:1. 5,000 images for validation (validation set), and 5,000 images for final testing (test set).
    - Batch size = 32.
    - SGD optimizer with an initial learning rate of 0.002.
    - Loss function: categorical cross entropy criterion.
    - Training iteration can be 90,000 or more (If you use epoch counting, epoch can be 58 or more). Perform a validation every 5000 iterations (3 epochs). It may take about 30 minutes on T4 GPU. If you don't have enough computational resources, you are allowed to reduce the number of images by taking a subset of this dataset or reduce the training iterations (epochs). State the size of your subset and iterations (epochs) in the README.
    - Use the default setting for the rest of the hyperparameters.
4. Plot the training loss, validation loss, and validation accuracy over the training iterations (or epochs). Fig. 1 shows an example. State whether the training appears to be overfitting and why.

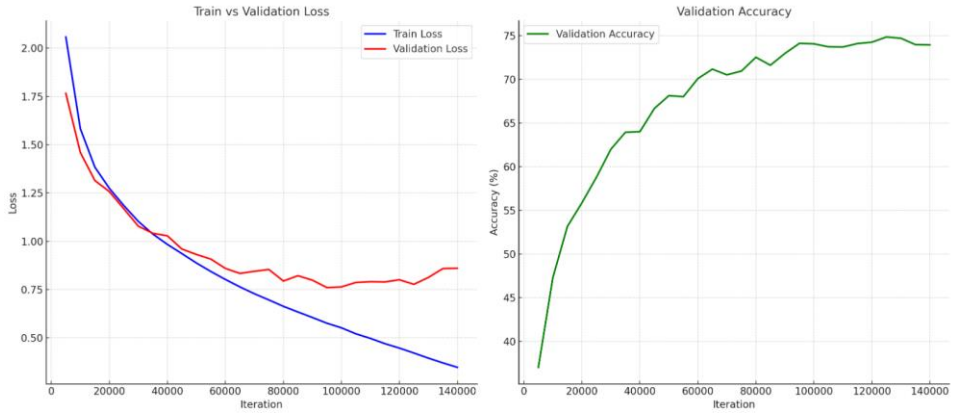

Fig. 1. Training loss, validation loss, and validation accuracy over the training iterations.

5. Please give the test accuracy on the test set from the iteration (or epoch) where the validation accuracy is maximum as your test accuracy result.
6. Let's discuss the effects of the Kernel size. Change all kernel sizes to 5x5 and train a new network with the same other hyperparameters. Compare the run time and the test accuracy of models under different kernel sizes and briefly discuss the possible factors that affect the performance of a CNN.
7. Use Pytorch Class *torchvision.models.resnet18* to implement a deep network ResNet18. Set the training iteration as 6000 or more (If you use epoch counting, epoch can be 5 or more) and perform a validation on the validation set every 500 iterations (1 epoch). Give the test accuracy on the test set from the iteration (or epoch) where the validation accuracy is maximum as the test accuracy result. The rest of the hyperparameters should be the same as the above shallow CNN. Note that:

   7.1 By setting the parameter *pretrained*, you can choose to either train a new ResNet18 model from scratch or fine-tune the ResNet18 model that has been fully trained on the ImageNet dataset.

   7.2 Since the image size of CIFAR10 is 32x32 and the standard ResNet18 accepts 224x224 input by default, we may need to first resize the input image to 224x224 (You are free to use other available transformation, such as padding). Besides, the output channel of the final fully connected layer of ResNet18 needs to be modified to 10 to meet the classification requirements of CIFAR10.

   Compare the impact of using a pre-trained ResNet18 versus not and discuss the reason.
   Compare the test accuracy of the deep ResNet18 versus the shallow CNN.

2 Part 2 - YOLOv8 Object Detection on Montréal Streets (30 points)

YOLOv8 is a state-of-the-art (SOTA) model for a wide range of object detection and tracking, instance segmentation, image classification and pose estimation tasks. In this section, you will be asked to take a photo of a street in Montreal and summarize the information in this image by using a trained YOLOv8 model. This section does not ask to implement and train the model from scratch. You can use a well-trained YOLOv8 model from its official implementation.

1. Use your cellphone or a digital camera to capture a street scene in Montréal.
2. Implement the trained YOLOv8 object detection model to identify what are the types of objects included in the image (such as person, bicycle, vehicle, tree) and count the number of each object.
3. Display the original and predicted images in your notebook.