# ELEC 576 / COMP 576 – Fall 2025
# Assignment 2

**Due:** November 6, 2025, 11:59 PMvia Canvas

## Submission Instructions

Every student must submit their report on Canvas in PDF format, providing intermediate and final results. You don't need to include temporary files such as Tensorboard logs in the submission. In this assignment, please submit your work according to the following instructions:

- Option 1: Submit all answers, screenshots, and pictures required together in one report in PDF format. Then include all relevant code and other files in a zip file (naming netid-assignment2.zip).
- Option 2: Convert your Jupyter Notebook to a PDF file and submit this PDF file with both the code and output. Make sure to run all the cells before submission. Don't forget to include textual answers to any written questions.

## GPU Resource

To accelerate the training using GPU, you can optionally use Amazon Web Services(AWS) GPU instance using AWS Education credits. You can also get additional AWS credits from Github Student Developer Pack. After having an AWS account, You can either create a fresh ubuntu instance and install software dependencies by yourself.
You can also use Google Colab for GPU resources.

## 1 Visualizing a CNN with CIFAR10

In this problem we will train a CNN on CIFAR10. After the network is trained, you will visualize the weights of the first convolutional layer to see what filters it learned. In addition, we will see what the activations look like by using the test images.

**a) CIFAR10 Dataset:** CIFAR10 is a dataset composed of natural images that represent 10 classes: *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, and *truck*. The dataset will be provided, and each image is grayscale and of size 28 x 28. Originally CIFAR10's images are RGB, and the original resolution is 32 x 32.

You have two options to import CIFAR10:

- Refer to this tutorial to import CIFAR10 with Pytorch.
- With `trainCifarStarterCode.py` (provided on Canvas) and the zipped folder, `CIFAR10`, you can import all the images, and you will need to perform additional preprocessing. You should use one-hot encoding for labels.

**b) Train LeNet5 on CIFAR10:** In this part you will implement a LeNet5 and train it on CIFAR10. You can read this online tutorial to get familiar with Pytorch implementation.
Below is the configuration of LeNet5. Please note that the original LeNet5 uses the tanh as the activation function after the convolutional and fully connected layers. In this assignment, you're welcome to experiment with different settings.

- Convolutional layer with kernel 5 x 5 and 6 filter maps followed by tanh
- Max Pooling layer subsampling by 2
- Convolutional layer with kernel 5 x 5 and 16 filter maps followed by tanh
- Max Pooling layer subsampling by 2
- Fully Connected layer that has input 5*5*16 and output 120
- Fully Connected layer that has input 120 and output 84
- Fully Connected layer that has input 120 and output 10 (for the classes)
- Softmax layer

Plot train/test accuracy and train loss. In addition, try to search for good hyper-parameters (e.g., training methods, learning rate, momentum values ...). **Hint:** Run a few epochs for each set of hyper-parameters and see how train/test accuracy and train loss change.

**c) Visualize the Trained Network:** Visualize the first convolutional layer's weights. They should look like Gabor filters (edge detectors). Figure 1 shows examples of these

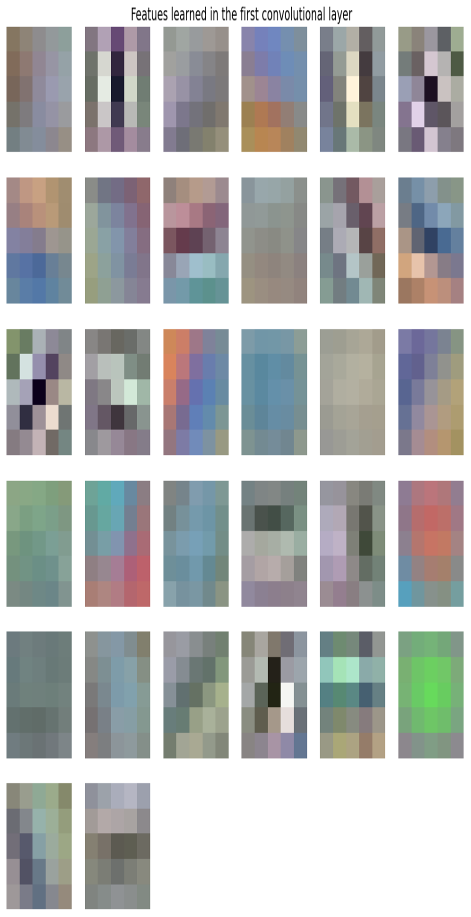

Figure 1: Filters learned in the first convolutional layer

filters. Notice that since you train your networks using gray-scale images, your filters will look slightly different. Also, show the statistics of the activations in the convolutional layers on test images.

## 2 Visualizing and Understanding Convolutional Networks

Read the paper Visualizing and Understanding Convolutional Networks by Matthew D. Zeiler and Rob Fergus. Summarize the key ideas of the paper.

**(Optional) Visualization of features in a fully trained model:** Apply one of the techniques discussed in the Visualizing and Understanding Convolutional Networks paper on the convnet trained in Problem 1. In the paper they used a validation data set, but in your case you would use the test set. Show your curiosity here :)

## 3 Build and Train an RNN on MNIST

An RNN is well-suited for tasks on time-series or sequential data. However, in this problem we will see how we can use RNN for object recognition and compare it to Assignment 1 where we used a CNN.

**a) Set up an RNN:** With a CNN, we would send in full images, yet with the RNN, inputs to the network are 28-pixel chunks of the images. For example, each row of the image will be sent to the network at each iteration. The starter code, `rnnMNISTStarterCode.py` (provided on Canvas), provides the details on how it splits the data so that it can work for the RNN. You should modify the following parameters in the starter code.

- Number of nodes in the hidden layer
- Learning rate
- Number of iterations
- Cost (hint: use softmax cross entropy with logits)
- Optimizer

**b) How about using an LSTM or GRU:** For example to use torch.nn.GRU and torch.nn.LSTM instead of RNN.

Plot the train/test accuracies and the train loss. What do you notice, and are the LSTM or GRU better? Also, change the number of hidden units and see how that affects the loss and accuracy.

**c) Compare against the CNN:** Compare with training using convnet in assignment 1 and describe any similarities or differences.

## Collaboration Policy

Collaboration both inside and outside class is encouraged. You may talk to other students for general ideas and concepts, but each student is expected to work on their write-up independently.

## Plagiarism

Plagiarism of any form will not be tolerated. You are expected to credit all sources explicitly.