1 CNNs and finetuning (40pt)

1. Download the CIFAR 10 dataset (original data can be found here, and here is a link to the pickled python version).

2. Use the pretrained Resnet18 model (from trochvision) to extract features. Use the features as inputs in a new multi-class logistic regression model (use nn.Linear/ nn.Module to define your model)

   (a) Describe any choices made and report test performance.

   (b) Display the top 5 correct predictions and the top 5 incorrect predictions in each class (show the images and the prediction labels) compactly.

3. Finetune the Resnet18 model's parameters suitably and repeat parts (a) and (b) from above. Compare the performance of finetuning versus using extracted features.

2 Movie embeddings (40pt)

Instead of embedding words, we will embed movies. In particular, if we can embed movies, then similar movies will be close to each other and can be recommended. This line of reasoning is analogous to the *distributional hypothesis of word meanings* (see `https://en.wikipedia.org/wiki/Distributional_semantics`). For words, this roughly translates to words that appear in similar sentences should have similar vector representations. For movies, vectors for two movies should be similar if they are watched by similar people.

Let the total number of movies be M. Let $X_{i,j}$ be the number of users that liked both movies i and j. We want to obtain vectors $v_1, ..., v_i, ..., v_j, ..., v_M$ for all movies such that we minimize the cost $c(v_1, ..., v_M) = \sum_{i=1}^{M} \sum_{j=1}^{M} \mathbf{1}_{[i \neq j]} (v_i^T v_j - X_{i,j})^2$. Here $\mathbf{1}_{[i \neq j]}$ is a function that is 0 when $i = j$ and 1 otherwise.

1. Compute data $X_{i,j}$ from the movielens (small) dataset (data and description). Briefly describe your data prep workflow (you can use *pandas* if needed).

2. Optimize function $c(v_1, ..., v_M)$ over $v_1, ..., v_M$ using gradient descent (using *pytorch* or *tensorflow*). Plot the loss as a function of iteration for various choices (learning rates, choice of optimizers etc).

3. Recommend top 10 movies (not vectors or indices but movie names) given movies (a) 'Apollo 13', (b) 'Toy Story', and (c) 'Home Alone' . Describe your recommendation strategy. Do the recommendations change when you change learning rates or optimizers? Why or why not?