Instructions:
Note that the Python files and trained models will be placed in the barn-shared folder three days after the deadline. The project currently contains five Python files, cnn_classification_cifar10.py, dataset.py, model.py, train_buildings.py, test.py

Below are the instructions for training the building model on CCI (currently).

```
cd scratch-shared
./conda init

//log in again

conda activate pytorch-env-shared

//schedule the code to run

srun -t 30 –gres=gpu:1 python train_buildings.py
```

```
// To test the pretrained model

srun -t 30 -gres=gpu:1 python test.py <path_to_data_dir> <name_of_label_file.csv>

//This program will load the trained model (the model we trained before submission) from the
models folder, then it will evaluate and print the accuracy on the data in the directory provided
```

Terminal Output:

Cnn_classification_cifar10.py (CIFAR10 Warmup) - Goal 70% test acc.

```
Train Epoch: 28 [10000/50000 (20%)]     Loss: 0.529506

Train Epoch: 28 [20000/50000 (40%)]     Loss: 0.668602

Train Epoch: 28 [30000/50000 (60%)]     Loss: 0.580766

Train Epoch: 28 [40000/50000 (80%)]     Loss: 0.510176
        Accuracy: 81.42%
Test set: Avg. loss: -91700.0211, Accuracy: 8121/10000 (81.21%)

Train Epoch: 29 [0/50000 (0%)]  Loss: 0.301322

Train Epoch: 29 [10000/50000 (20%)]     Loss: 0.375155

Train Epoch: 29 [20000/50000 (40%)]     Loss: 0.514007

Train Epoch: 29 [30000/50000 (60%)]     Loss: 0.410843

Train Epoch: 29 [40000/50000 (80%)]     Loss: 0.659053
        Accuracy: 81.29%
Test set: Avg. loss: -92216.8068, Accuracy: 8240/10000 (82.40%)

Train Epoch: 30 [0/50000 (0%)]  Loss: 0.426450

Train Epoch: 30 [10000/50000 (20%)]     Loss: 0.589516

Train Epoch: 30 [20000/50000 (40%)]     Loss: 0.514605

Train Epoch: 30 [30000/50000 (60%)]     Loss: 0.504464

Train Epoch: 30 [40000/50000 (80%)]     Loss: 0.543723
        Accuracy: 82.09%
Test set: Avg. loss: -91286.1939, Accuracy: 8202/10000 (82.02%)
(pytorch-env-shared) [RNL2fmrd@npl41 barn]$
```

Train_buildings.py (CNN Model) - Goal 90% test acc.

```
Train Epoch: 30 [2900/4380 (66%)]       Loss: 0.323846

Train Epoch: 30 [3000/4380 (68%)]       Loss: 0.251844

Train Epoch: 30 [3100/4380 (70%)]       Loss: 0.310559

Train Epoch: 30 [3200/4380 (73%)]       Loss: 0.261890

Train Epoch: 30 [3300/4380 (75%)]       Loss: 0.296492

Train Epoch: 30 [3400/4380 (77%)]       Loss: 0.234192

Train Epoch: 30 [3500/4380 (80%)]       Loss: 0.180947

Train Epoch: 30 [3600/4380 (82%)]       Loss: 0.392068

Train Epoch: 30 [3700/4380 (84%)]       Loss: 0.364467

Train Epoch: 30 [3800/4380 (86%)]       Loss: 0.227045

Train Epoch: 30 [3900/4380 (89%)]       Loss: 0.254229

Train Epoch: 30 [4000/4380 (91%)]       Loss: 0.312469

Train Epoch: 30 [4100/4380 (93%)]       Loss: 0.262813

Train Epoch: 30 [4200/4380 (95%)]       Loss: 0.347567

Train Epoch: 30 [3440/4380 (98%)]       Loss: 0.245557
        Accuracy: 90.53%
Test set: Avg. loss: 80.0020, Accuracy: 459/485 (94.64%)
(pytorch-env-shared) [RNL2fmrd@npl41 barn]$
```

Questions:

1) How many parameters does your network have? Please provide your calculations, layer by layer. You can ignore batch normalization parameters – just count the convolution and fully connected ones.

The network has 4,594,287 parameters. The calculations to determine this parameter count can be seen below.

Parameter Calculation for CNN Model

Convolutional Layers

- Conv1 = 3×3×3×16+16=448
- Conv2 = 3×3×16×32+32 = 4,640
- Conv3 = 3×3×32×64+64 = 18,496
- Conv1 + Conv2 + Conv3 = 448 + 4640 + 18496 = 23,584

Batch Normalization Layers

- Bn1 = 16×2 = 32
- Bn2 = 32×2 = 64
- Bn3 = 64×2 = 128
- Bn1 + Bn2 + Bn3 = 32 + 64 + 128 = 224

Fully Connected Layers

- 64×(height/8)×(width/8)
- Height = 189/8 = 23, Width = 252/8 = 31
- Fc1 = 64×23×31×100+100 = 4,563,300
- Fc2 = 100×64+64 = 6464
- Fc3 = 64 × 11 + 11 = 715
- Fc1 + Fc2 + Fc3 = 4,563,300+6464+715 = 4,570,479

Total Parameters

Total Parameters=Conv1+Conv2+Conv3+Bn1+Bn2+Bn3+FC1+FC2+FC3
Total Parameters=23,584+224+4,570,479 = 4,594,287

2) How did you improve the CIFAR10 accuracy? What was the issue?

We improved CIFAR10 accuracy by applying data augmentation on the training data and adding four layers to the CNN_small class instead of one. The augmentations include random horizontal flips, rotation, crop, and color jitter. The dataset was also converted to a tensor and normalized before training. The reasoning for these augmentations is to increase training data diversity, generating a more complete and thorough training model.