CNN Quiz Summary

The CNN quiz will ask you to

1. obtain a fresh copy of the pytorch tutorial,
2. apply a series of steps, and
3. report your accuracy statistics after each step.

Description of Steps

Each quiz question will ask you to perform a combination of the following steps, and then report your accuracy. You do not need to perform all the following steps in their order here.

Step A1: Obtaining a Fresh Copy of the Assignment

- If you do not want to lose your current progress in the assignment, look for the following question ***"How can I get a fresh copy of the assignment while preserving the old one?"*** question in the **FAQ about Jupyter Notebooks** article.
- Please note that **you should not use or build upon your submitted notebook for this quiz, and you need to start from scratch** for this quiz.

Step A2: Copying the PyTorch Tutorial

- You can copy and paste the content from https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html into your jupyter notebook one cell at a time just like you did when you started working on the assignment.
- You only need to do this for the CIFAR portion of the assignment.
- Upon finishing the copy & paste operation, locate the following two lines (under the cell titled **"1.1 Loading the Data"**)

  ```
  trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                          download=True, transform=transform)

  testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                         download=True, transform=transform)
  ```

  and modify their `root` argument as the following:

  ```
  root_data = '/home/jovyan/work/release/CNN-lib/data_cifar'
  trainset = torchvision.datasets.CIFAR10(root=root_data, train=True,
                                          download=True, transform=transform)

  testset = torchvision.datasets.CIFAR10(root=root_data, train=False,
                                         download=True, transform=transform)
  ```

- **Important Note**: Make sure you copy every single piece of evaluation or visualization code, even if it had nothing to do with the training. This is important so that you do not have a hard time reproducing the results.
- **Important Note**: Here is a print of a notebook with the tutorial material copied into it. Make sure that (1) you have all the cells this template has, and (2) the cell contents are identical before proceeding to the next part.

Step A3: Controlling the Randomized Effects

Copy and paste the following 3 lines of code right at the top of the cell titled **"1.1 Loading the Data"**:

```
np.random.seed(12345)
random.seed(12345)
torch.manual_seed(12345)
```

After pasting these 3 lines, your cell should look like the following

```
np.random.seed(12345)
random.seed(12345)
torch.manual_seed(12345)

message = 'You can implement the pre-processing transformations, data sets, data loaders, etc. in this cell. \n'
message = message + '**Important Note**: Read the "Grading Reference Pre-processing" bullet above, and look at the'
message = message + ' test pre-processing transformations in the "Autograding and Final Tests" section before'
message = message + ' training models for long periods of time.'
print(message)

classes = ('plane', 'car', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

...
```

Why are we doing this?

- For maximum reproducibility and better comparison, it is often a good idea to fix your random number generator seeds to a fixed number.
- This way you can achive "deterministic stochasticity", meaning that while you did not specify the random numbers in your code, your code will generate the same exact output every time you run it from scratch.
- That being said, trying to exploit the statistics by finding a random seed that increases your accuracy is an unforgivable sin.

Step A4: Restart the Python Kernel

- On the top left of your screen, there is a "Kernel " menu. Click on it.
- Click on either the "Restart" or the "Restart and Clear Output" options, and wait a few seconds for the restart request to go through.
- Sometimes the restart request does not execute properly or require more time to finish.
  - Do not perform this step too quickly.
  - If you see a small circle filled with black next to the "Python 3 (Threads: 2)" message in the top right corner of your screen, this means that the kernel is busy executing your request, and you need to wait for the circle to be unfilled.

Step A5: Running the CIFAR Portion of the Notebook

- Start running the notebook cells one-by-one from the top of the notebook.
- Make sure that no cell raises a pytho error, and the trained network gets stored at `cifar_net.pth`.
  - You may want to back-up your old `cifar_net.pth` file that you used for your assignment submission, but we will not ask you for it.
- Run all the cells before the MNIST part starts, including the cell titled "**1.8 Autograding and Final Tests**"

Step A6: Reporting the Final Training Loss and Overall Testing Accuracy

- After running the CIFAR part of the notebook without an error, the cell titled "**1.8 Autograding and Final Tests**" will print a final testing accuracy value at the end of its output.
- Look for the phrase "Overall Testing Accuracy" as the following:

  ```
  __________
  Overall Testing Accuracy: XX.YY %%
  ```

- The number `XX.YY` will be your final testing accuracy that should be reported in your quiz.
- For the final training loss, look for the output of the training cell (i.e., the cell under the title **"1.5 Training the Model"**). For example, suppose the following is the entire training log:

  ```
  [1,  2000] loss: 2.222
  [1,  4000] loss: 1.869
  [1,  6000] loss: 1.679
  [1,  8000] loss: 1.588
  [1, 10000] loss: 1.511
  [1, 12000] loss: 1.445
  [2,  2000] loss: 1.400
  [2,  4000] loss: 1.363
  [2,  6000] loss: 1.335
  [2,  8000] loss: 1.297
  [2, 10000] loss: 1.292
  [2, 12000] loss: 1.345
  Finished Training
  ```

  The value `1.345` is the last printed loss, and you are supposed to enter this value as your final training loss.

Step A7: Updating the Pre-processing transformations

Locate the following lines from the pytorch tutorial in the cell titled **"1.1 Loading the Data"**

```
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
```

and change it to the following

```
transformation_list = [transforms.RandomAffine(degrees=30, translate=(0.01, 0.01),
                                               scale=(0.9, 1.1), shear=None,
                                               resample=0, fillcolor=0),
                       transforms.ToTensor(),
                       transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]

transform = transforms.Compose(transformation_list)
```

This way you will be using a matching pre-processing for both the training and testing phases.

Step A8: Increasing the Number of Optimization Epochs

Locate the following `for` loop under the title "**1.5 Training the Model**"

```
for epoch in range(2):  # loop over the dataset multiple times
```

and change the number of epochs to 10:

```
for epoch in range(10):  # loop over the dataset multiple times
```

- It is worth noting that only fixing the pre-processing transformation (Step A6) and training the model for 80 epochs (i.e., for about an hour and half) would have gotten a full score upon submitting the assignment.

Step A9: Increasing the Second Convolutional Layer's Channels

Locate the following Neural Network constructor lines

```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        ...
```

and increase the number `16` to `64` in both `self.conv2` and `self.fc1` definition lines:

```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 64, 5)
        self.fc1 = nn.Linear(64 * 5 * 5, 120)
        ...
```