Instructions for homework submission

1. Complete two parts in this homework:
   - Math questions: Include your solution in LaTeX document. Show your work. Submission with embedded photos of handwritten work will not be accepted.
   - Programming questions: Complete the given skeleton `Python` code. **For questions requiring visualization, analysis, and discussion, please include your solution in the same LaTeX document.**
2. Submit your work to Gradescope including:
   - A PDF document for written parts: `FirstName_LastName_HW3.pdf`. LaTeX source code is not required.
   - A completed `Python` code: `FirstName_LastName_HW3.py`.
   - **A pretrained model file:** `model.pt`
   - **There are two separate submission portals on** `Gradescope`**:** one for code and one for the report.
   - Please assign your answer in PDF report to its corresponding question when submitting to Gradescope.
   - Failure to follow submission instruction will result in loss of marks.
3. Start early!
4. Total: 100 points.

If you find that training your model is taking too long or if you require additional computational power, you are encouraged to explore the High Performance Research Computing (HPRC) resources available at Texas A&M University. For more information and to access these resources, please visit `https://hprc.tamu.edu/`.

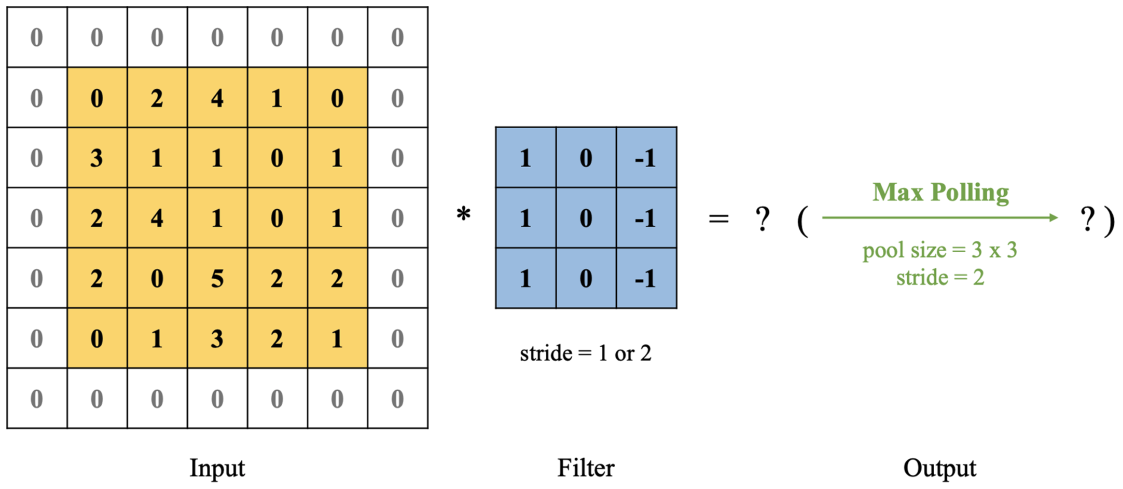
This project utilizes a subset of the SUN397 database[1][2]. We acknowledge and appreciate their efforts in collecting and maintaining this data.

[1] Xiao, Jianxiong, et al. "Sun database: Large-scale scene recognition from abbey to zoo." 2010 IEEE computer society conference on computer vision and pattern recognition. IEEE, 2010.

[2] Xiao, Jianxiong, et al. "Sun database: Exploring a large collection of scene categories." International Journal of Computer Vision 119 (2016): 3-22.

Question 1: Convolution Operation

In this problem, we will use the convolution operation on the matrix using the 3x3 filter as shown below.



Apply the convolution operation for all the following settings respectively, and write your outputs in your LaTeX report.

- Without max pooling and with a stride of 1.
- Without max pooling and with a stride of 2.
- With max pooling and with a stride of 1.

Question 2: Image Classification using CNN

In this homework, you will implement two primary components:

1. A PyTorch Dataset class, `SUN397Dataset`, designed to load images from a given folder structure. You are asked to implement the `SUN397Dataset` class such that:
   - It inherits from `torch.utils.data.Dataset`.
   - It performs necessary transformation (i.e. normalization) and assigns an integer label (starting at 0) based on the class name (ordered alphabetically). *(Note: normalization for test data will be handled by autograder. You can verify your computed mean/std is correct via testcase 2.2.)*
   - Its constructor accepts a single path to your data folder, for example `./data/train`.
   - The `__getitem__` method correctly returns a tuple `(image_tensor, label_tensor)`.
   - The `__len__` method returns the total number of samples in your dataset.
2. A Convolutional Neural Network (CNN) model, `CNN`, to classify images into the appropriate classes. You have to use at least **ONE** convolutional layer (i.e., `nn.Conv2d`).

You will then train your CNN on a subset of the SUN397 dataset, save your best trained model to `model.pt` and aim for **75%** classification accuracy on our reserved test set.

Best Practices & Tips

None of these are mandatory, but you might find them helpful along the way!

1. **Check your dataset:** Verify that iterating over your `SUN397Dataset` returns `(image, label)` pairs with the expected shapes. Debugging issues here first will help you avoid errors in training. You can create a validation split to better understand your model behavior.
2. **Network architecture:** Perform a parameter count to identify which components of your network hold the most parameters, and analyze whether they are contributing effectively to the learning process.
3. **Hyperparameters:** Tweak schedulers, optimizers, and other hyperparameters to improve your model.
4. **Data augmentation:** You may incorporate additional transforms (random cropping, flipping, etc.) during training to generalize better.
5. **Argparse:** An example of an argument parser is provided to you, aiming to improve your hyperparameter tuning efficiency.

Important Constraints

- You are **NOT** allowed to use pretrained model from internet. Your submitted model weights need to be learned from scratch.
- **DO NOT** attempt to manipulate your training data in a way that gives you an unfair advantage.

Failure to follow these constraints will disqualify your submission.