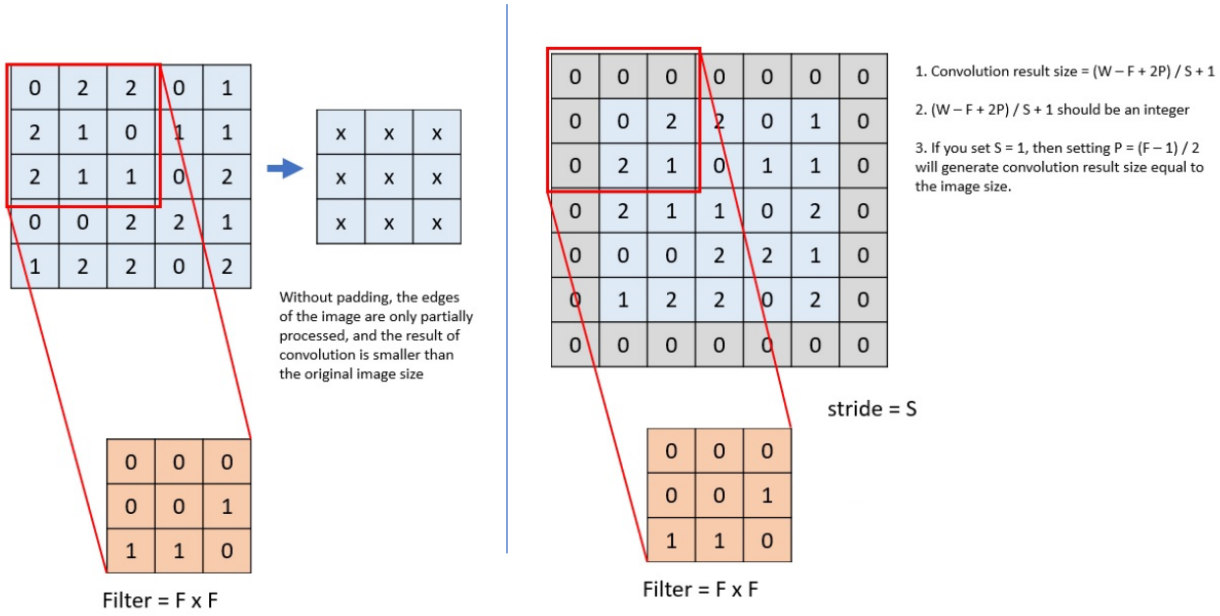
Homework assignment 2
Due: Thu., Mar. 10, 2022, 11:59pm

Problem 1 (8 points). When applying a CNN to a tensor, we have to be aware of the relationship between the tensor size, kernel size, padding size around the tensor, and the distance the filter moves (the stride) during convolution.
- Without padding (i.e., padding size = 0), the pixels on the perimeter of the input tensor are only partially processed. The result of convolution will have a smaller size than the original tensor size.
- When computing the convolution, we start with the convolution at the top-left corner of the input tensor and then slide it over both down and to the right (i.e., stride = 1). But sometimes, either for computational efficiency or because we wish to down-sample, the kernel is moved by more than one pixel at a time, skipping the intermediate elements. The number of rows and columns traversed are referred as the *stride.*



Please find the outputs under the two settings in the above figures. (Hint: the convolutional results under both settings are 3-by-3 matrices).
- left: padding = 0 and stride = 1;
- right: padding =1 and stride = 2.

Problem 2 (12 points). Transfer learning. Please study the enclosed "2_Transfer_learning.ipynb". One can take advantage of the GPU resource of Google Colab to run the code (One needs to use your CCID to login and configs your GPU environment by "Runtime->change runtime type->GPU"). Running the code should get validation accuracy above 97%. Note, in the transfer learning code, the base model ResNet50 is pre-trained on ImageNet and frozen in this task. Only the top classification layer is trained using the cat&dog image set. You task is to read the code, and then modify the code so that the whole model (base mode+classifier) are finetuned. Please upload your code and compare the evaluation accuracy

after 10 epoch training. What is your training, validation, and evaluation accuracy? Is your finetuned model overfitted?
Hint 1: It is critical to only do fine-tuning ***after*** the classifier with the frozen backbone model has been trained to convergence. Suppose you mix randomly-initialized trainable layers with trainable layers that hold pre-trained features. In that case, the randomly-initialized layers will cause very large gradient updates during training, which will destroy your pre-trained features.

Hint 2: It's also critical to use a very small learning rate at this stage, because you are training a much larger model than in the first round of training. With a relatively small cat&dog dataset, you are at risk of overfitting very quickly if you apply large weight updates. Here, you only want to readapt the pre-trained weights incrementally.

Comment: In Assignment 1, we are exposed to the python code based on Pytorch. Since Tensorflow is another widely used deep learning framework, the provided code in Assignment 2 uses Tensorflow as the backend support. As a deep learning person, one should be familiar with both frameworks and able to read/write codes with them.