Practice with CNNs

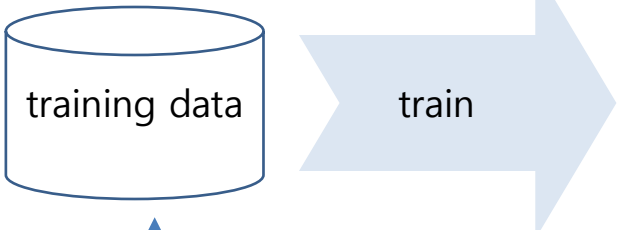
Machine Intelligence Lab
Handong Global University

CNN 실습

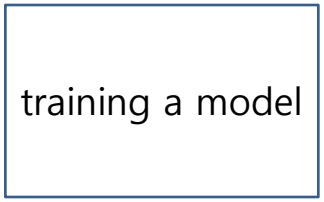
시간	코드 실습 내용
10:30 – 10:40	실습 소개 및 코드 실행
10:40 – 11:00	ML Pipeline PyTorch CNN library
11:00 – 11:40	CNN for MNIST: 코드 설명
11:40 – 12:00	**실습 및 휴식:** **Practice:** change training params epoch, batch_size
점심시간	
13:00 – 13:30	CNN for CIFAR10: - Implement forward - Implement conv layer
13:30 – 13:50	**실습 및 휴식:** **Practice:** change model architecture conv_layer, learning rate
13:50 – 14:30	CNN for my own image - Load my own image - Data augmentation

Workflow for supervised learning

- training



train



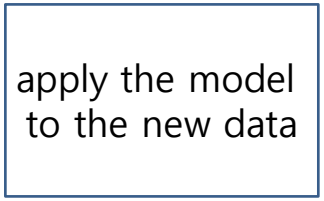

```
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
```

trained model

```
import Torchvision.datasets
import Torchvision.transforms
```

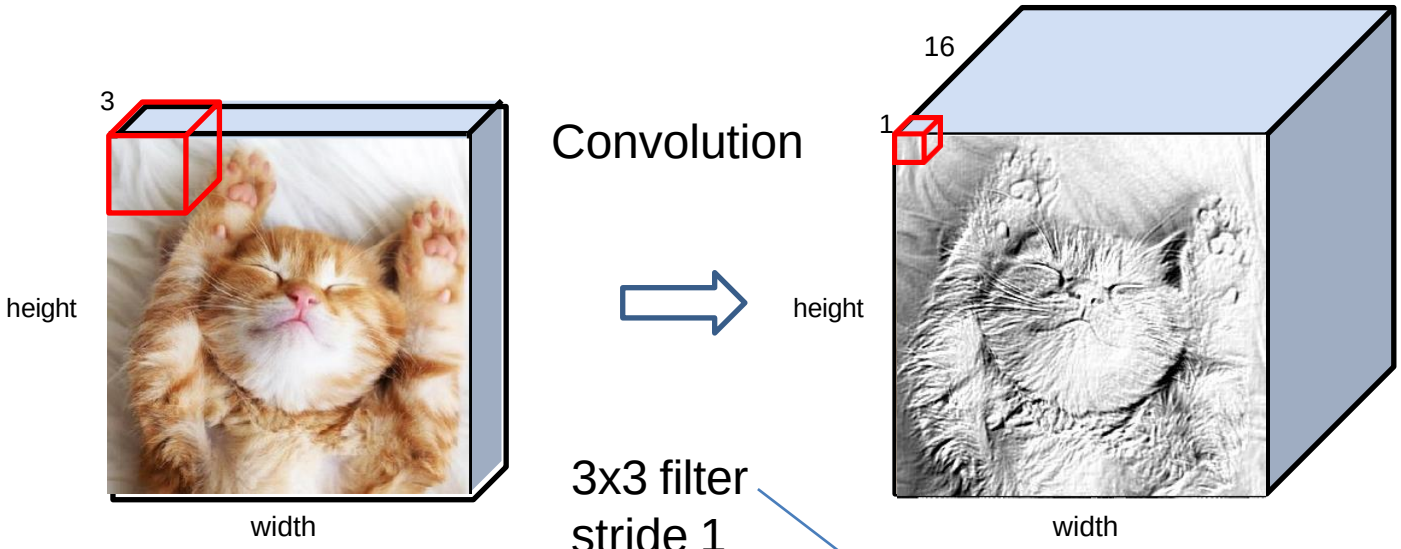
- testing

test



prediction result

PyTorch Library: convolution layer (step 1)

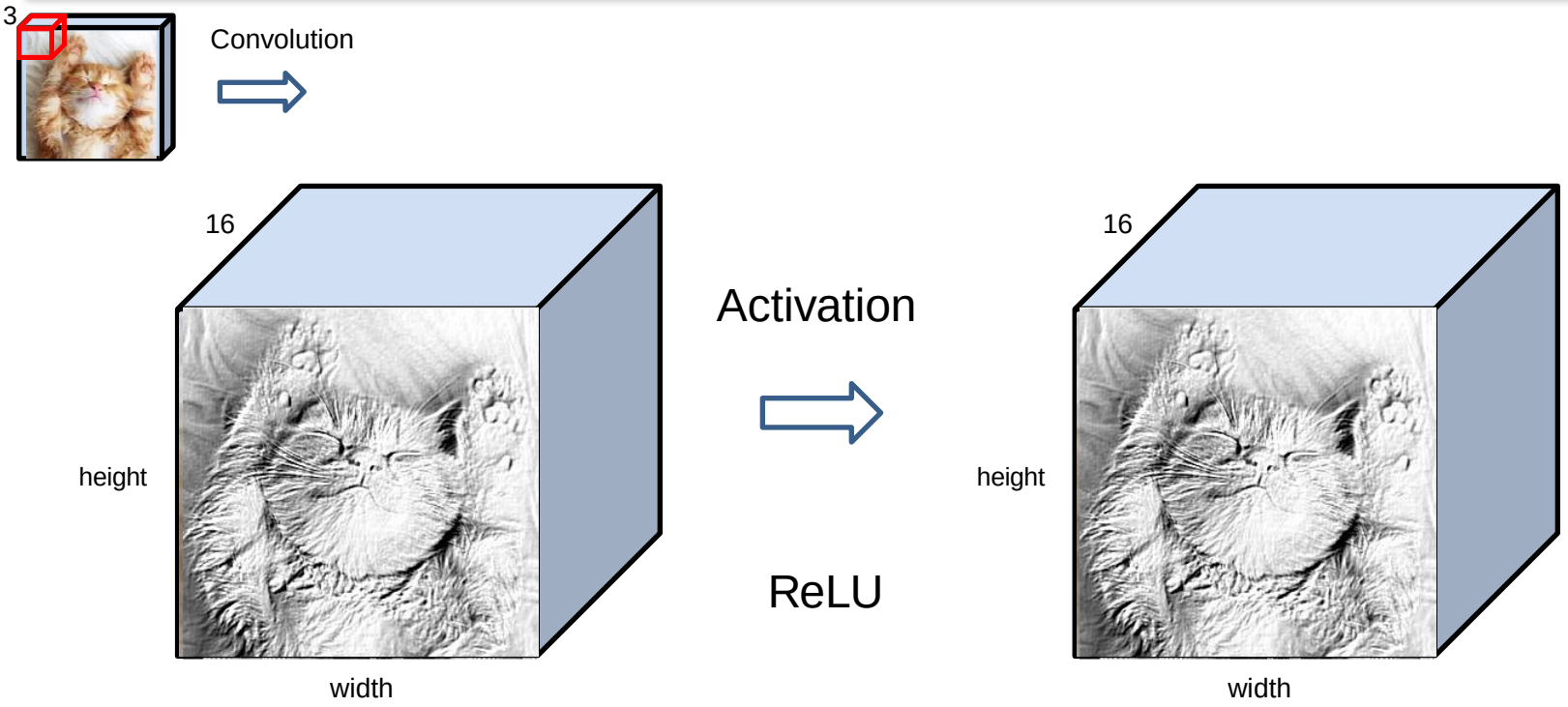



```
CLASS torch.nn.Conv2d(in_channels: int, out_channels: int, kernel_size: Union[T, Tuple[T, T]], stride: Union[T, Tuple[T, T]] = 1, padding: Union[T, Tuple[T, T]] = 0, dilation: Union[T, Tuple[T, T]] = 1, groups: int = 1, bias: bool = True, padding_mode: str = 'zeros')
```

[SOURCE]

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(1, 20, 5, 1),
```

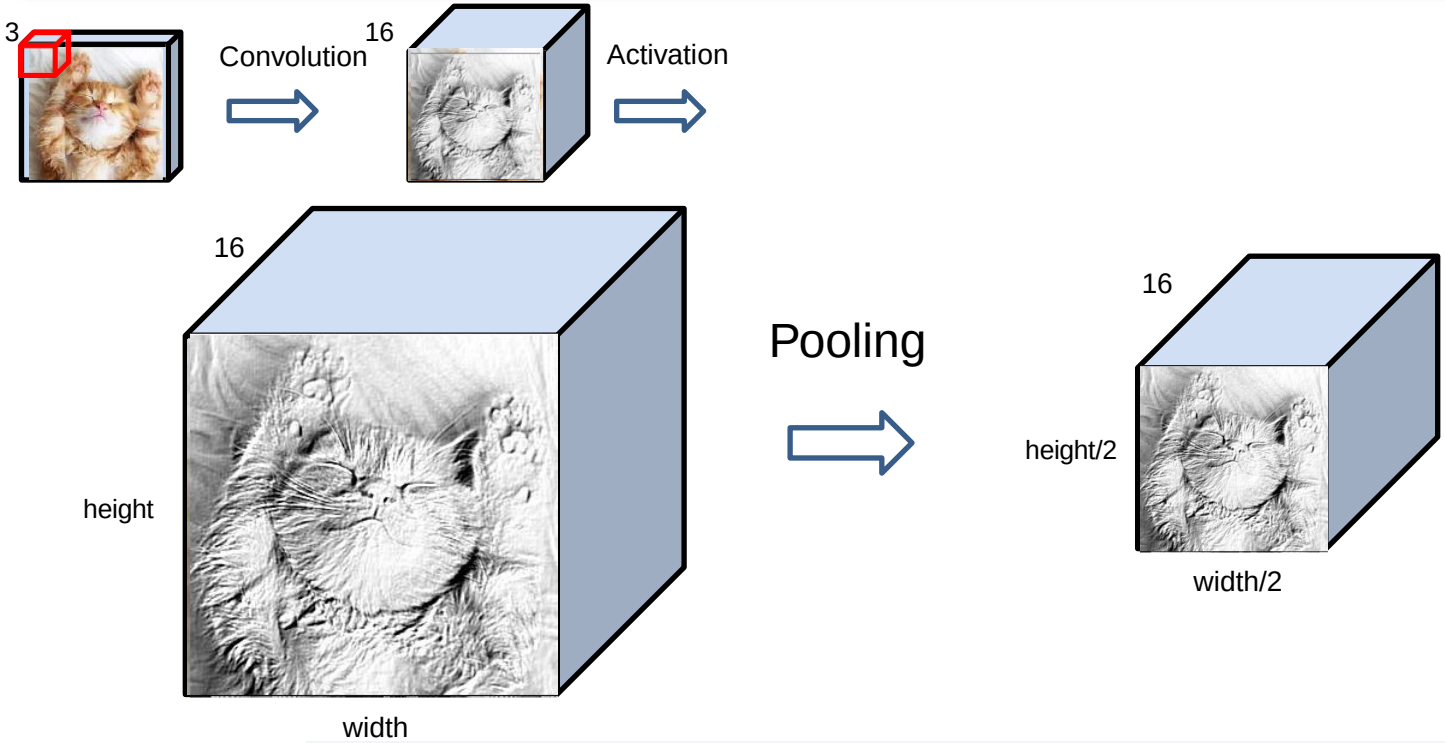
PyTorch Library: convolution layer (step 2)



CLASS `torch.nn.ReLU(inplace: bool = False)` [SOURCE]

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(1, 20, 5, 1),
            nn.ReLU(),
```

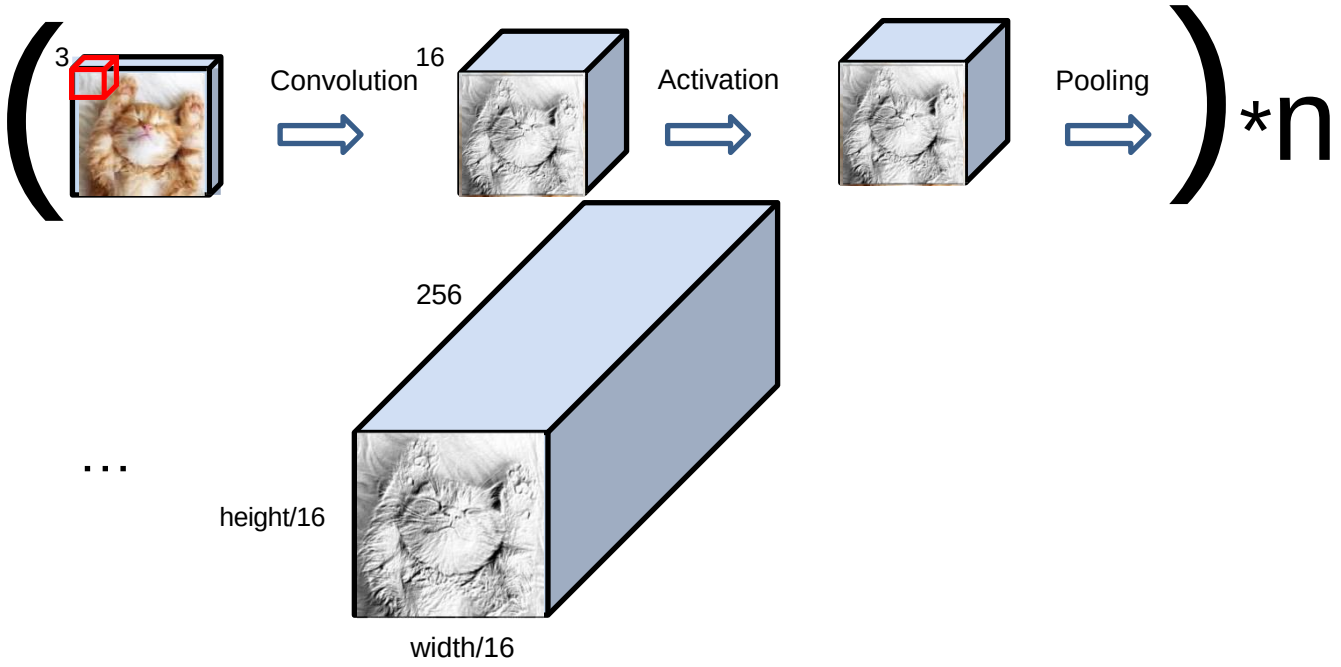
PyTorch Library: convolution layer (step 3)



CLASS torch.nn.MaxPool2d(*kernel_size: Union[T, Tuple[T, ...]], stride: Optional[Union[T, Tuple[T, ...]]] = None, padding: Union[T, Tuple[T, ...]] = 0, dilation: Union[T, Tuple[T, ...]] = 1, return_indices: bool = False, ceil_mode: bool = False*) [SOURCE]

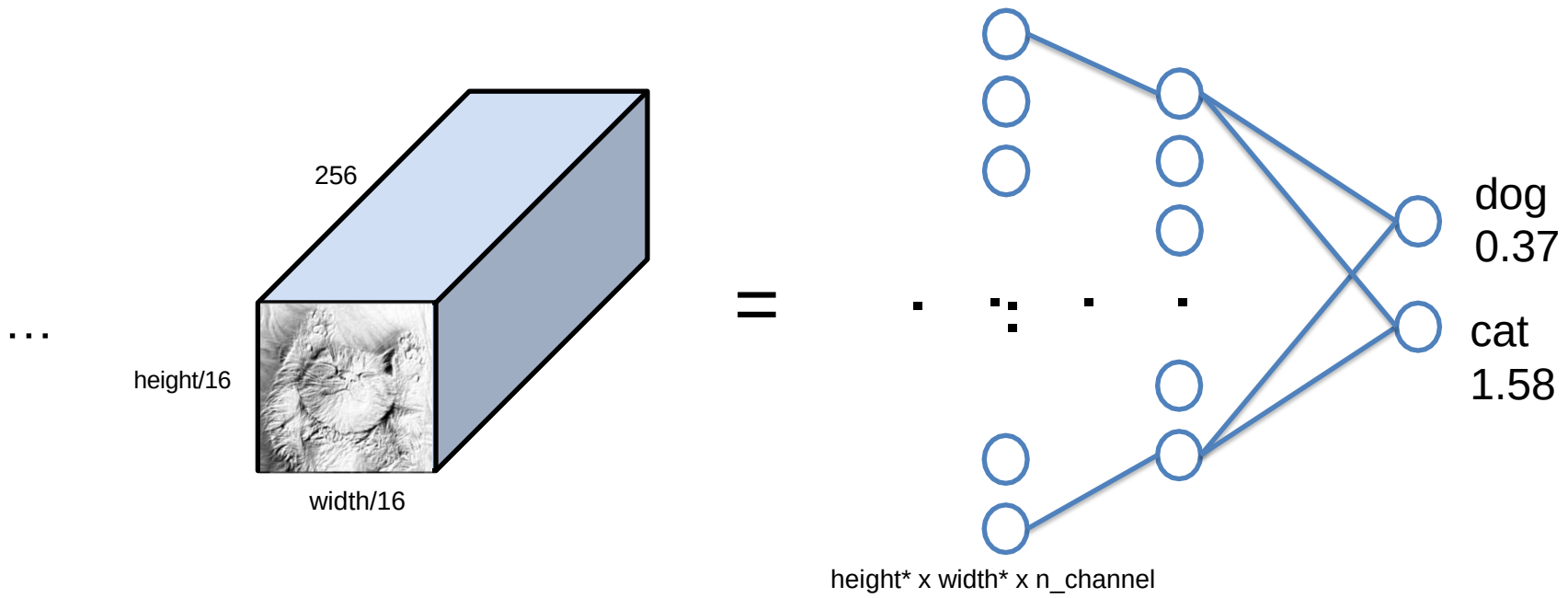
```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(1, 20, 5, 1),
            nn.ReLU(),
            nn.MaxPool2d(2,2),
```

iterative convolution layer



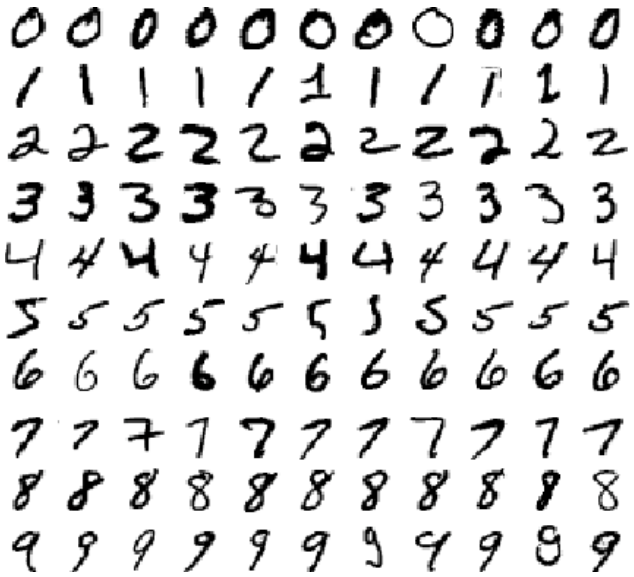

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(1, 20, 5, 1),
            nn.ReLU(),
            nn.MaxPool2d(2,2),
            nn.Conv2d(20, 50, 5, 1),
            nn.ReLU(),
            nn.MaxPool2d(2,2)
        )
```

fully connected layer




```
        self.fc = nn.Sequential(
            nn.Linear(conv_size, 500), # conv_size = 4*4*50
            nn.Linear(500, 10)
        )
```

Practice1 – MNIST



MNIST dataset [28X28] = [1 X 784]

Hand Written Digit Number from 0 to 9
Data includes data and label.

Pytorch Dataset(torchvision) provides
60,000 images to train,
10,000 images to test.

Practice1 – MNIST

Train the model with different training parameters

1. Train the model with large number of epochs
   - Try 30 epochs
   - Try to modify test() to return test accuracy
   - Try to plot ‘Test Accuracy’ for each epoch using **Matplotlib**

2. Try different batch_size
   - Try 15 epochs
   - Default batch_size is 32
   - Try batch_size=64 and batch_size=16
   - Try plotting ‘Test Accuracy’ for each epoch using **Matplotlib**

Practice2 – CIFAR10

10 classes

airplane

automobile

bird

cat

deer

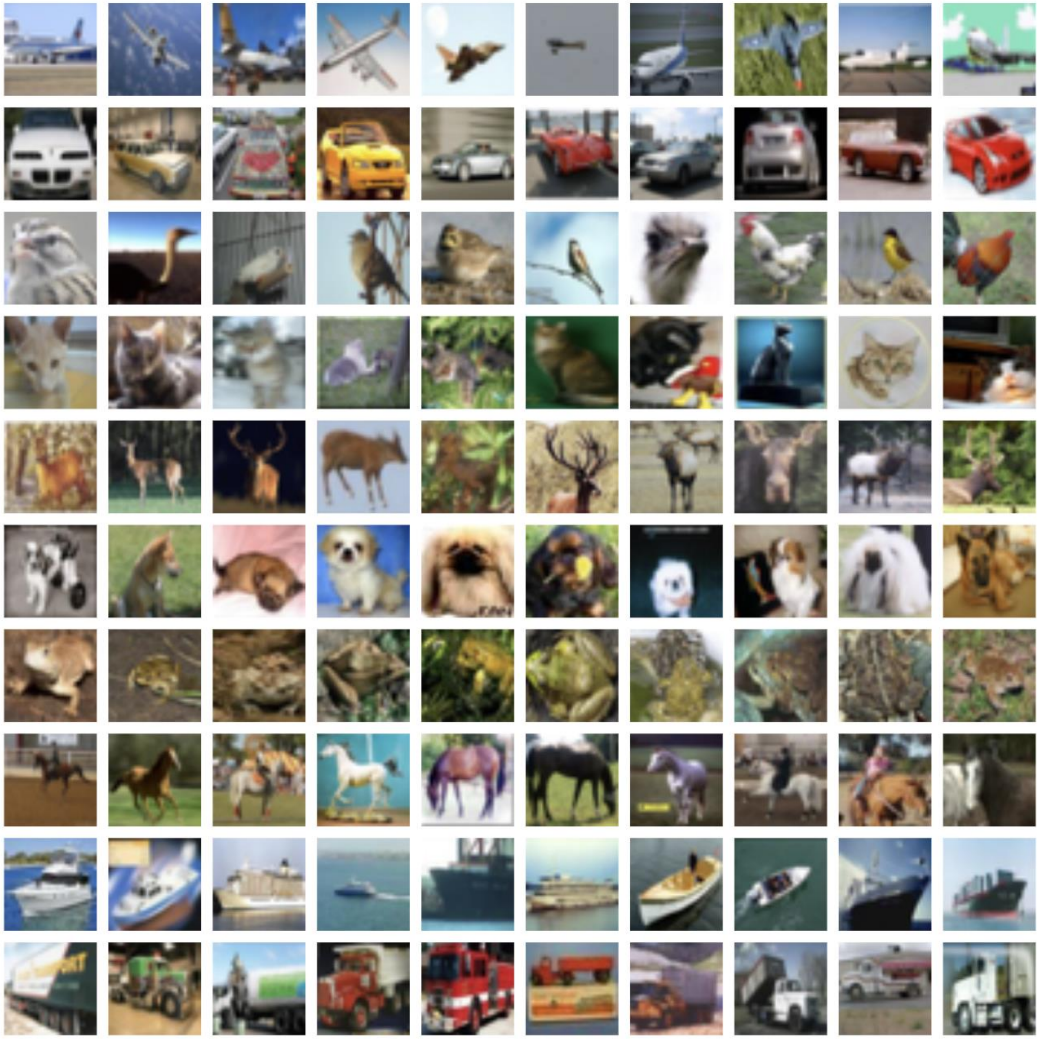
dog

frog

horse

ship

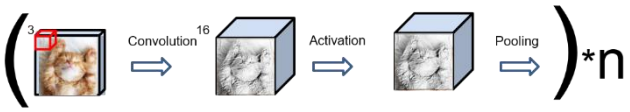
truck



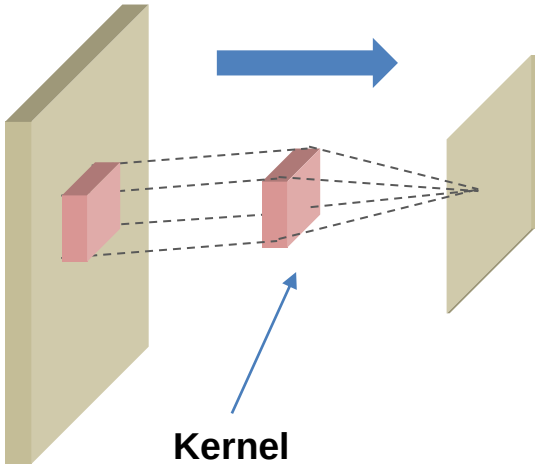
Practice2 – CIFAR10

Change model architecture and learning rate

1. Add 2 **conv + ReLU** layers and 1 **Maxpooling** layer
   - CIFAR10_CNN_Model will have 6 conv layers
   - Observe whether model's performance increase



2. Try different **kernel sizes**
   - Default is k=3
   - Try k=1 and k=5



Performance can be reported with Accuracy

Practice3 - Training your own data example

- put images in the class directories for train, valid and test.
- for example, with 'HDH' and 'OH' classes
  - ./drive/My Drive/public/train/HDH/*.jpg
  - ./drive/My Drive/public/train/OH/*.jpg
  - ./drive/My Drive/public/valid/HDH/*.jpg
  - ./drive/My Drive/public/valid/OH/*.jpg

Practice3 - HGU Dataset

- 5 Buildings in Handong Global University (HGU)


NTH


ANH


HDH


Hyoam

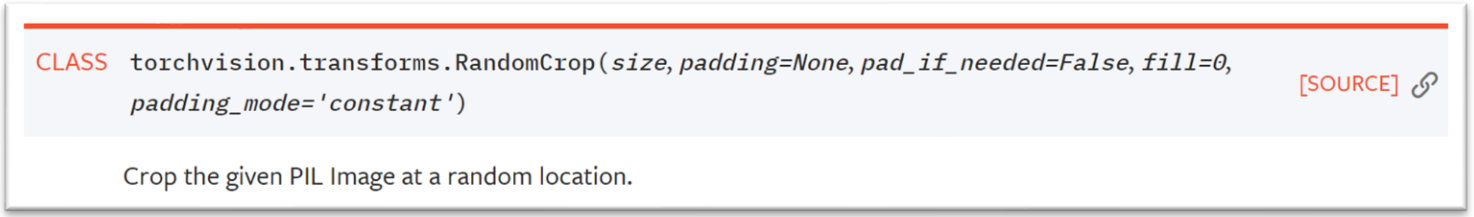

OH

- Dataset Size
  - Train: 4,186
  - Validation: 241
  - Test: 200
  - Extra: 41 images
- Image Size and Channel
  - (120 * 80), (256 * 192), (256 * 341), and so on
  - RGB (3 channels)

Practice3 - HGU Dataset

1. Try data augmentation method

Use RandomCrop() in torchvision.Transforms class. Observe model performance



CLASS `torchvision.transforms.RandomCrop(size, padding=None, pad_if_needed=False, fill=0, padding_mode='constant')` [SOURCE]

Crop the given PIL Image at a random location.

2. Try different optimizers

- Default is Adam with lr=0.001
- Try SGD with lr=0.01 and momentum=0.9
- Try RMSprop with lr=0.001

Performance can be reported with valid accuracy and test accuracy

Transfer Learning

- We don't have a huge dataset
- Some imagenet images look similar with our images
- Pretrained models on imagenet can be useful for training our model

Our data, just 3k

Imagenet, 1.2Million

Machine Intelligence Lab
https://milab.handong.edu/

Handong Global University