PATTERN RECOGNITION USING PYTHON

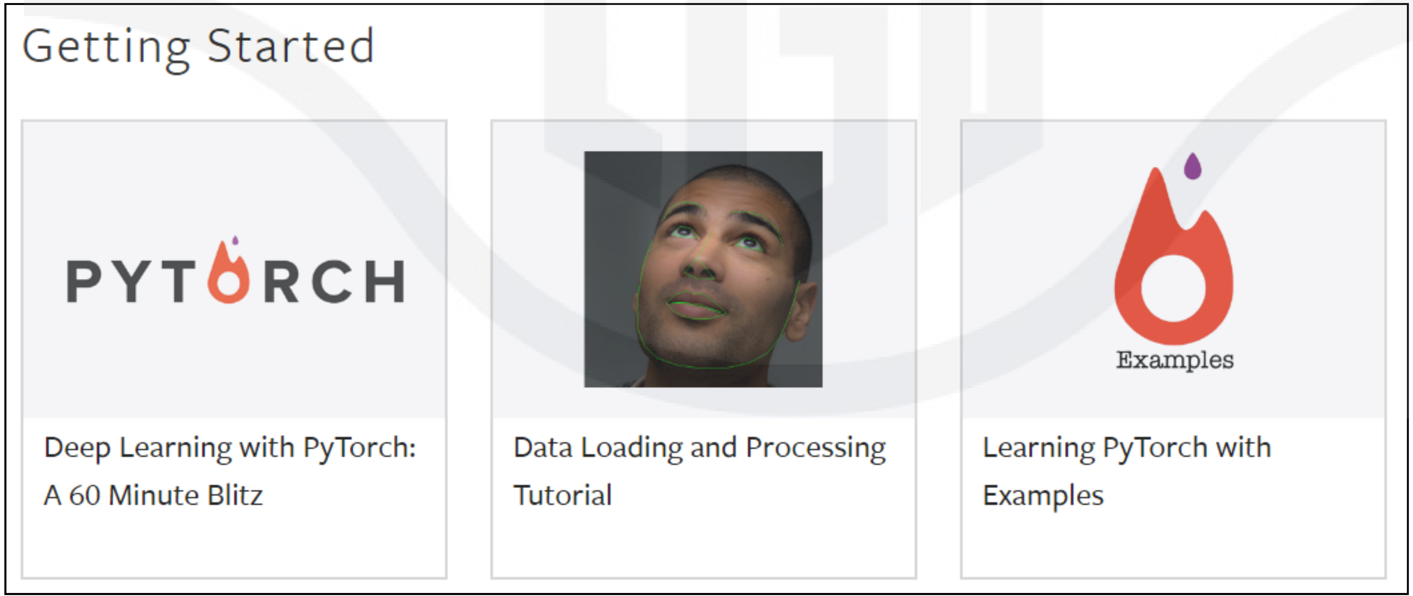
PyTorch

Wen-Yen Hsu
Dept Electrical Engineering
Chang Gung University, Taiwan

2019-Spring

PyTorch Introduction (Deep Learning Framework)

- Deep integration into Python allows popular libraries and packages to be used for easily writing neural network layers in Python
- A rich ecosystem of tools and libraries extends PyTorch and supports development in computer vision, NLP and more.



Getting Started

Deep Learning with PyTorch: A 60 Minute Blitz

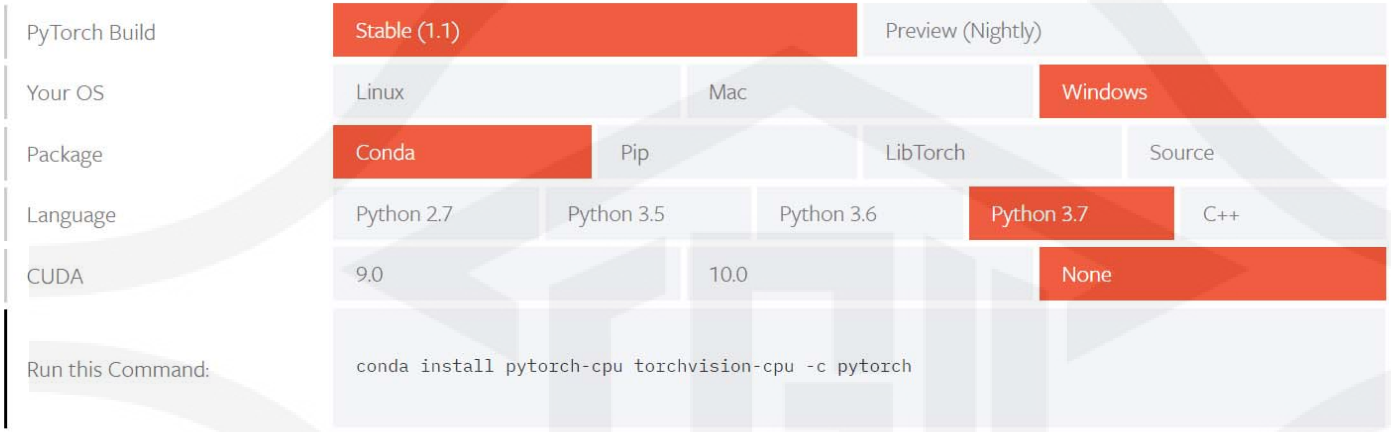
Data Loading and Processing Tutorial

Learning PyTorch with Examples

https://pytorch.org/tutorials/

PyTorch Intstall

- Using Anaconda without GPU (Anaconda Prompt)

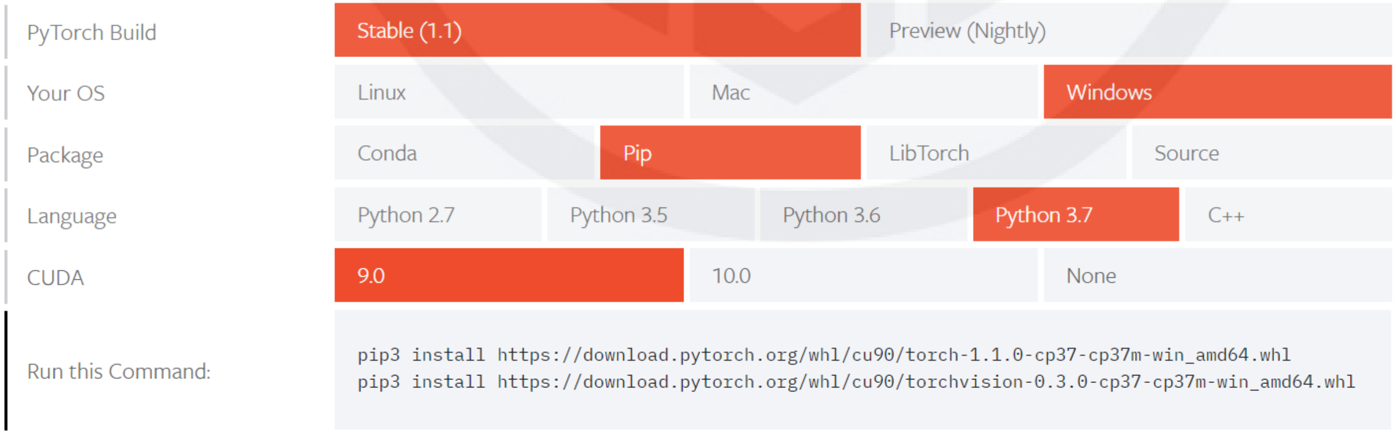


PyTorch Build	[✓ Stable (1.1)]	Preview (Nightly)			
Your OS	Linux	Mac	[✓ Windows]		
Package	[✓ Conda]	Pip	LibTorch	Source	
Language	Python 2.7	Python 3.5	Python 3.6	[✓ Python 3.7]	C++
CUDA	9.0	10.0	[✓ None]		

Run this Command:

```
conda install pytorch-cpu torchvision-cpu -c pytorch
```

- Using VScode with NV GPU (Terminal)



PyTorch Build	[✓ Stable (1.1)]	Preview (Nightly)			
Your OS	Linux	Mac	[✓ Windows]		
Package	Conda	[✓ Pip]	LibTorch	Source	
Language	Python 2.7	Python 3.5	Python 3.6	[✓ Python 3.7]	C++
CUDA	[✓ 9.0]	10.0	None		

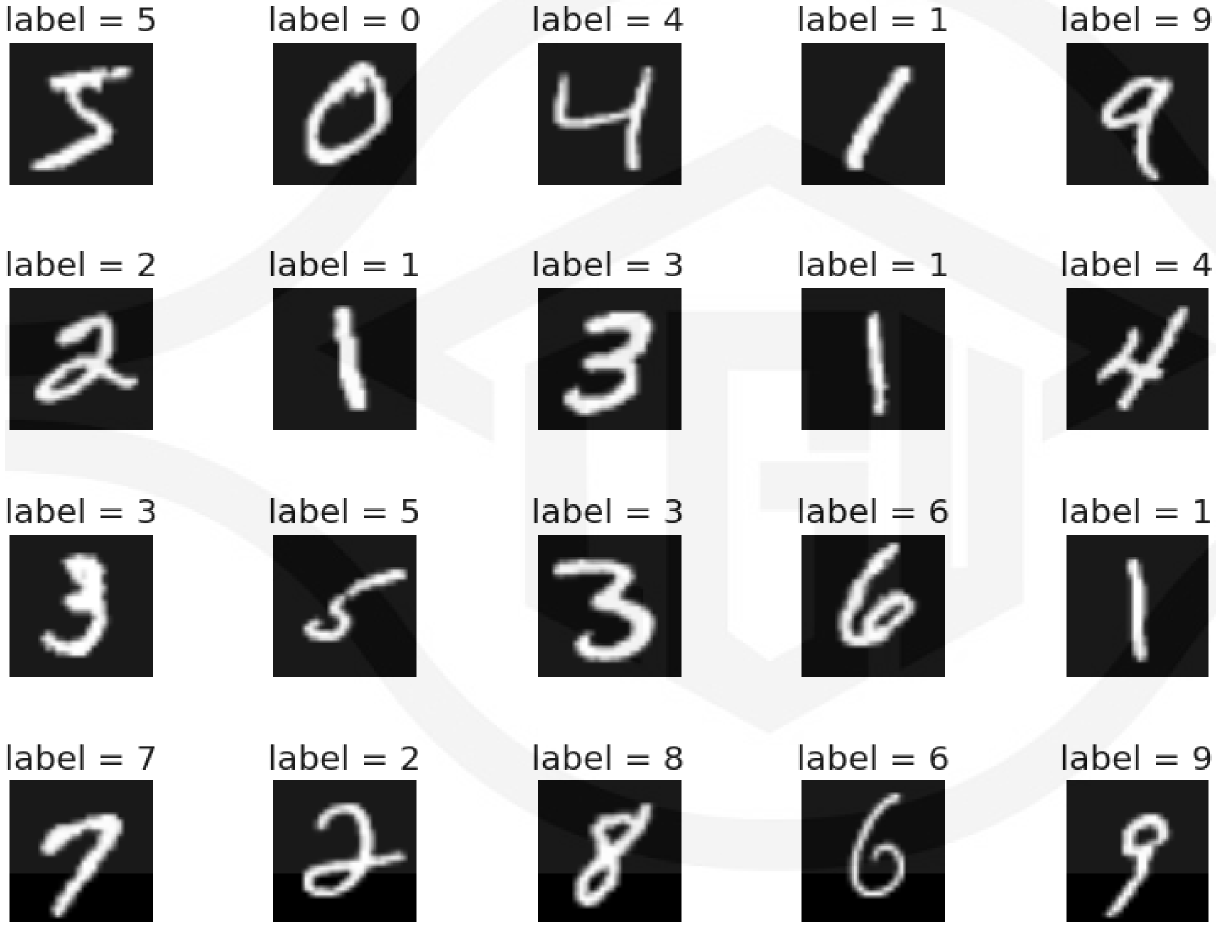
Run this Command:

```
pip3 install https://download.pytorch.org/whl/cu90/torch-1.1.0-cp37-cp37m-win_amd64.whl
pip3 install https://download.pytorch.org/whl/cu90/torchvision-0.3.0-cp37-cp37m-win_amd64.whl
```

Work Flow

- Build neural network
  - Through by (torch.nn.Module)
- Read data
- Transform format & preprocessing
  - Work in (torch.tensor)
- Dataset encapsulation
- Initial model
  - Choose loss function (torch.nn.CrossEntropyLoss)
  - Using optimizers (torch.optim.SGD)
- Training phase
- Testing phase

Handwritten Digits



http://yann.lecun.com/exdb/lenet/

Build Neural Network

- Using MLP as sample

```python
import torch as t
import torch.nn as nn
import torch.nn.functional as F
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.hidden1 = nn.Linear(784, 512)
        self.hidden2 = nn.Linear(512, 128)
        self.output = nn.Linear(128, 10)
    def forward(self, x):
        x = F.relu(self.hidden1(x))
        x = F.relu(self.hidden2(x))
        x = self.output(x)
        return x
```

Read Data (pandas)

- Load MNIST data

```
## Load dataset
df = pd.read_csv('mnist_784.csv', header=0)
y = df.iloc[:, -1].values
print(y.shape)
X = df.iloc[:, 0:-1].values
print(X.shape)
```

Transform Format & Preprocessing (sklearn pytorch)

- Prepare for next step, encapsulation

```
## Preprocessing
X = X / 255.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=1, stratify=y)
## Put numpy array to tensor
X_train_tensor = t.tensor(X_train, dtype=t.float,
device=cpu)
y_train_tensor = t.tensor(y_train, dtype=t.long,
device=cpu)
X_test_tensor = t.tensor(X_test, dtype=t.float,
device=cpu)
y_test_tensor = t.tensor(y_test, dtype=t.long,
device=cpu)
```

Dataset Encapsulation

- Using Pytorch API to create data loader

```
from torch.utils import data
from torch.utils.data import DataLoader

## Use dataLoader
torch_dataset = data.TensorDataset(X_train_tensor,
y_train_tensor)
loader = DataLoader(dataset=torch_dataset,
batch_size=batch, shuffle=True, num_workers=0)
```

Initial Model

- Make the model available

```
## Initial model
model = Net()
print(model)
optimizer = t.optim.SGD(model.parameters(),
lr =learning_rate)
loss_func = t.nn.CrossEntropyLoss()
```

Training Phase

- Apply batch trianing

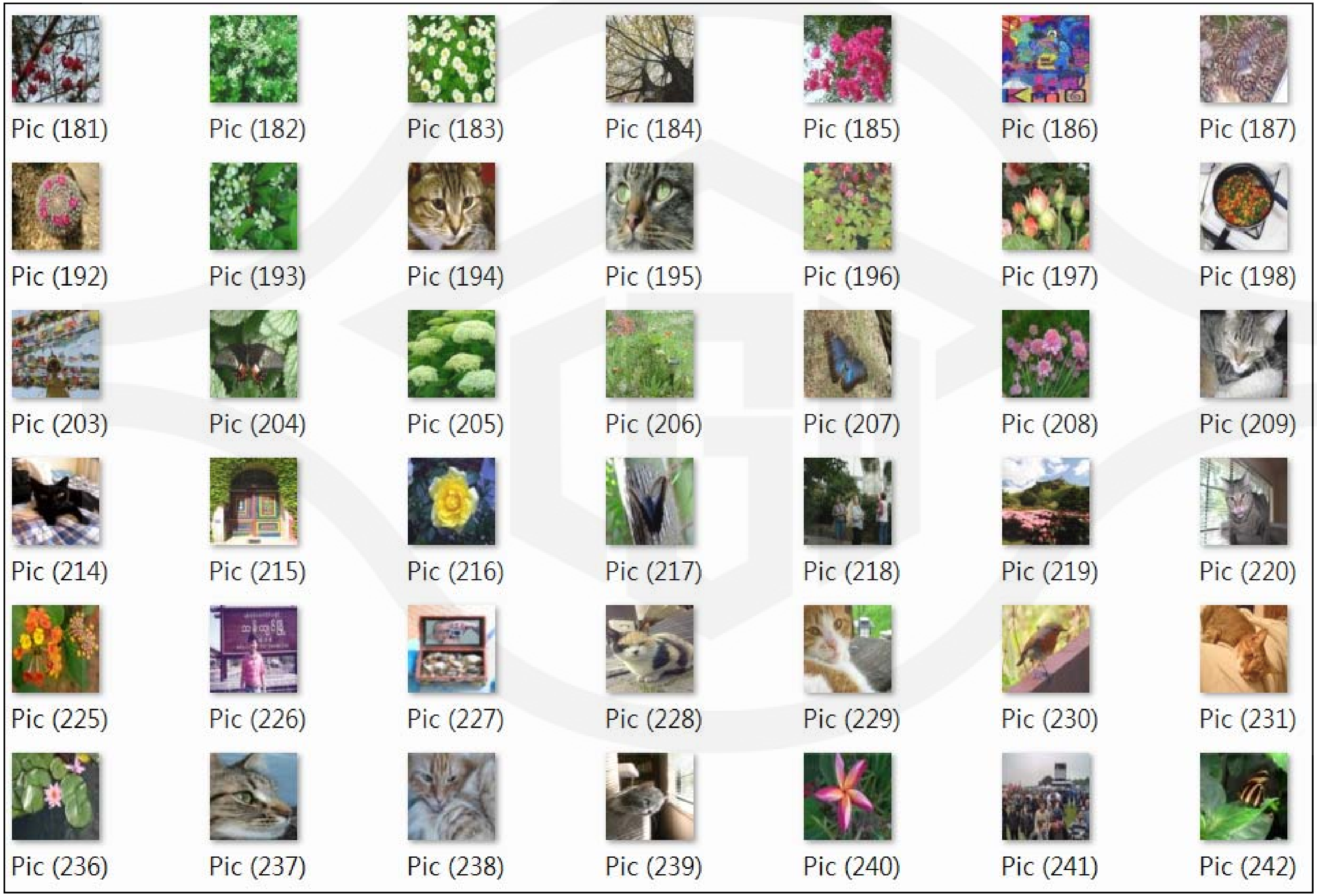
```
for epoch in range(epoch_time):
    loss_average = np.zeros(1)
    for step, (batch_x, batch_y) in enumerate(loader):
        optimizer.zero_grad()
        prediction = model(batch_x)
        loss = loss_func(prediction, batch_y)
        loss.backward()
        optimizer.step()
        loss_cpu = loss.cpu().data.numpy()
        loss_average = np.add(loss_average,
loss_cpu/batch)
    print('Epoch=', epoch)
    print('Loss=%.4f' % loss_average)
```

Testing Phase

- Benchmark model performance with test set

```
y_test_hat_tensor = model(X_test_tensor)
y_test_hat = y_test_hat_tensor.data.numpy()
## change float to index
y_test_hat = np.argmax(y_test_hat, axis=1)
print(y_test_hat)
print(y_test)
print("Test set score: %f" % accuracy_score(y_test,
y_test_hat))
```

Recognize Cats (RGB Picture)

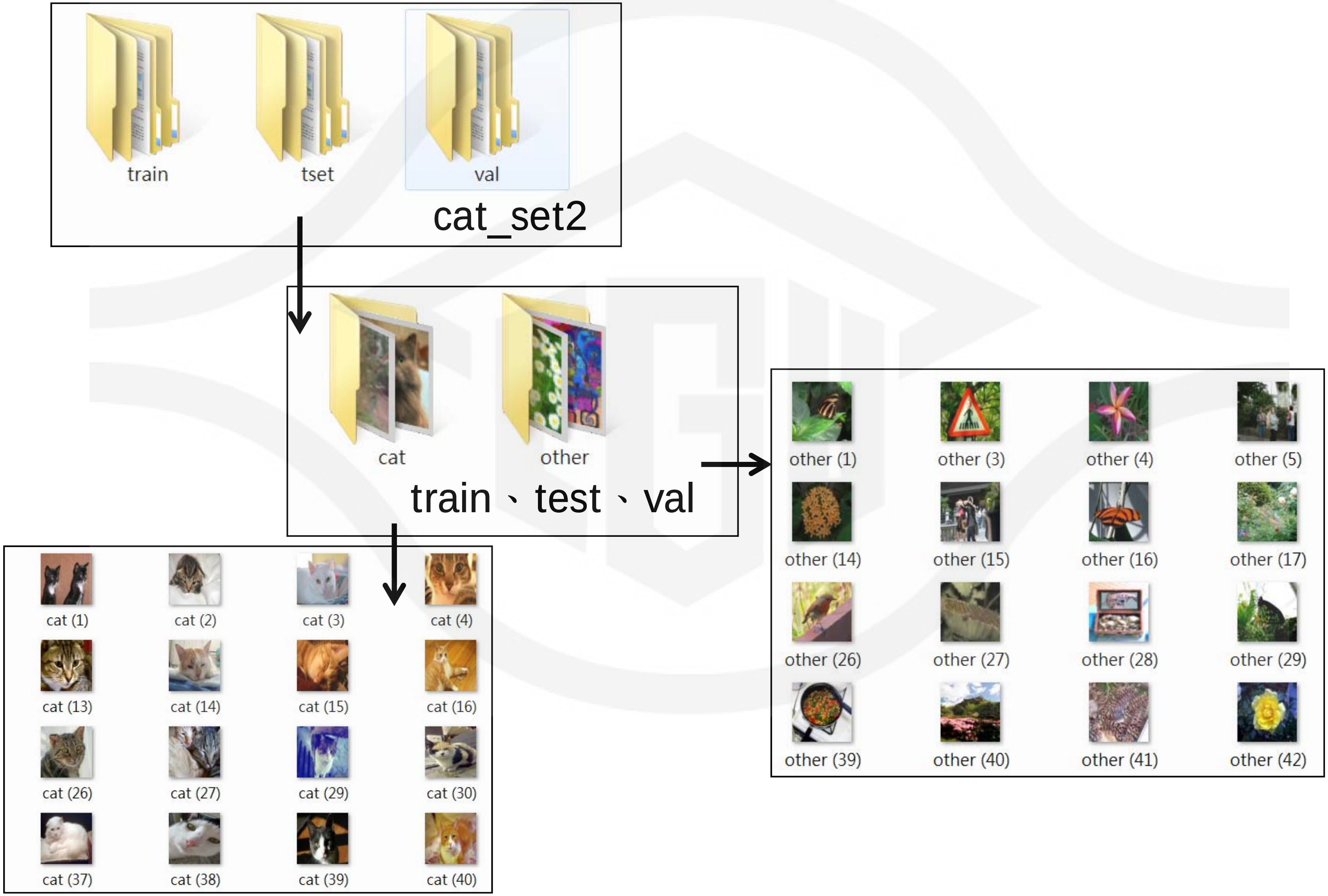


Work Flow in PyTorch

- Build neural network
- Read data
- Transform format & preprocessing
- Dataset encapsulation
- Initial model
- Training phase
- Testing phase

Replace by Torchvision API

Put pictures into folders in a specific format



cat_set2

train 、test 、val

Torchvision API

- Deal with RGB images

```
from torchvision.datasets import ImageFolder
from torchvision import transforms as T
transform = T.Compose([
    T.Resize(pic_size),
    T.CenterCrop(pic_size),
    T.ToTensor(),
    T.Normalize(mean=[.5, .5, .5], std=[.5, .5, .5]) ])
train_dataset = ImageFolder('D:/cat_set2/train', transform=transform)
print(train_dataset.class_to_idx)
print('train set num', len(train_dataset.imgs))
train_loader = DataLoader(train_dataset, batch_size=batch,
                    shuffle=True, num_workers=0,
                    drop_last=False)
val_dataset = ImageFolder('D:/cat_set2/val', transform=transform)
print('validation set num', len(val_dataset.imgs))
val_loader = DataLoader(val_dataset, batch_size=len(val_dataset.imgs),
                    shuffle=True, num_workers=0)
test_dataset = ImageFolder('D:/cat_set2/tset', transform=transform)
print('test set num', len(test_dataset.imgs))
test_loader = DataLoader(test_dataset, batch_size=len(test_dataset.imgs),
                    shuffle=True, num_workers=0)
```

Reference

- Vishnu Subramanian. Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch. Packt Publishing, 2018.
- https://pytorch.org/get-started/locally/