Kissipo Learning for Deep Learning

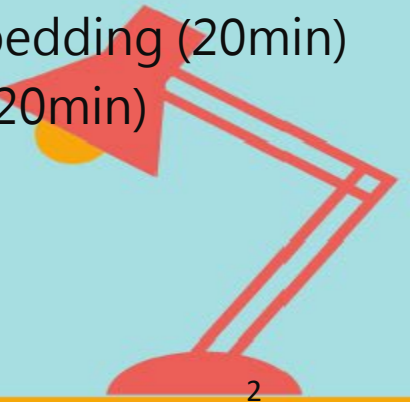
Topic 18: AOI simple Pipeline (A) (20min)

Hsueh-Ting Chu

KLDL-W7-T18

Topics

- Topic 01: Introduction to Deep Learning (20min)
- Topic 02: Kissipo Learning for Deep Learning (20min)
- Topic 03: Python quick tutorial (20min)
- Topic 04: Numpy quick tutorial (15min)
- Topic 05: Pandas quick tutorial (15min)
- Topic 06: Scikit-learn quick tutorial (15min)
- Topic 07: OpenCV quick tutorial (15min)
- Topic 08: Image Processing basics (20min)
- Topic 09: Machine Learning basics (20min)
- Topic 10: Deep Learning basics (20min)
- Topic 11: TensorFlow overview (20min)
- Topic 12: CNN with TensorFlow (20min)
- Topic 13: RNN with TensorFlow (20min)
- Topic 14: PyTorch overview (20min)
- Topic 15: CNN with PyTorch (20min)
- Topic 16: RNN with Pytorch (20min)
- Topic 17: Introduction to AOI (20min)
- Topic 18: AOI simple Pipeline (A) (20min)
- Topic 19: AOI simple Pipeline (B) (20min)
- Topic 20: Introduction to Object detection (20min)
- Topic 21: YoloV5 Quick Tutorial (20min)
- Topic 22: Using YoloV5 for RSD (20min)
- Topic 23: Introduction to NLP (20min)
- Topic 24: Introduction to Word Embedding (20min)
- Topic 25: Name prediction project (20min)



Course Schedule

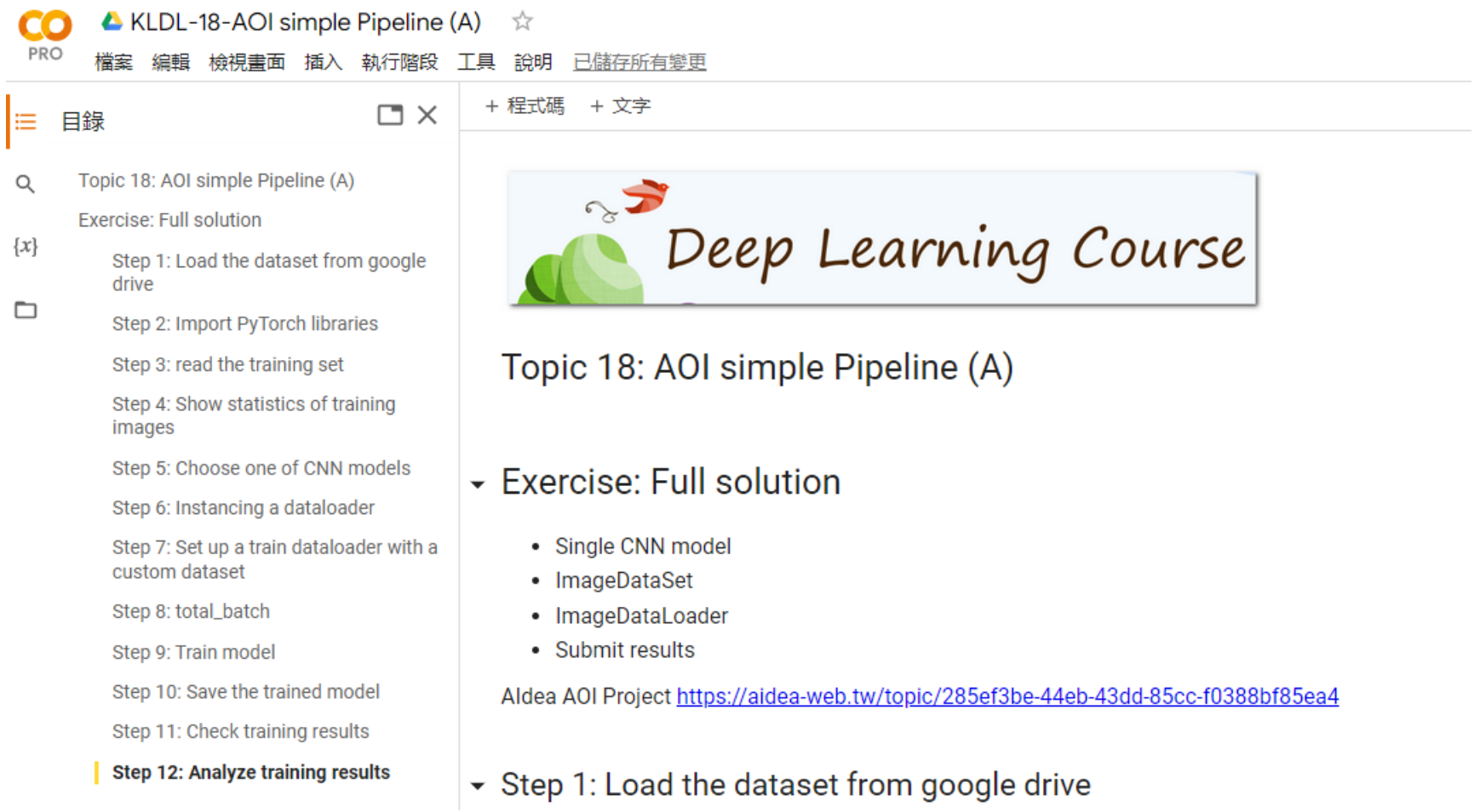
- W1 - Course Introduction
- W2 - DL Programming Basics(1)
- W3 - DL Programming Basics(2)
- W4 - DL with TensorFlow
- W5 - Midterm
- W6 - DL with PyTorch
- W7 - AOI hands-on project
- W8 - RSD hands-on project
- W9 - NLP hands-on project
- W10 - Final exam

DL: Deep Learning
AOI: Automated Optical Inspection
RSD: Road Sign Detection
NLP: Natural Language Processing

Week 7 Topics

- Topic 17: Introduction to AOI (20min)
- Topic 18: AOI simple Pipeline (A) (20min)
- Topic 19: AOI simple Pipeline (B) (20min)

AOI pipeline (A)



Step 1: Load the dataset from google drive

KLDL-18-AOI simple Pipeline (A)

檔案 編輯 檢視畫面 插入 執行階段 工具 說明 已儲存所有變更

+ 程式碼 + 文字

Exercise: Full solution

- Single CNN model
- ImageDataSet
- ImageDataLoader
- Submit results

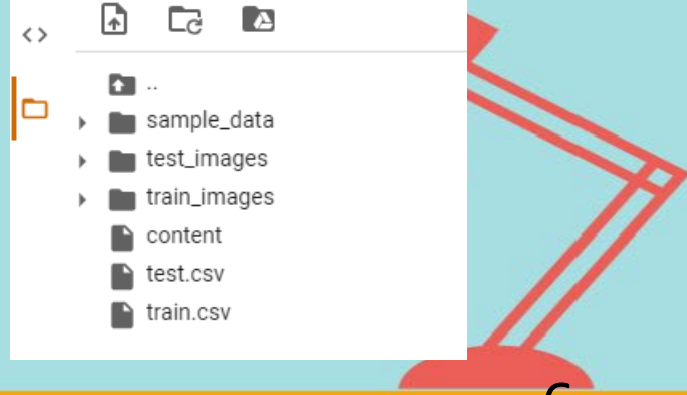
AIdea AOI Project https://aidea-web.tw/topic/285ef3be-44eb-43dd-85cc-f0388bf85ea4

Step 1: Load the dataset from google drive

If the following command does not work, please download it, put it on your Google drive, and set up sharing

Download from: https://drive.google.com/file/d/1tovCO2gsjesjJ8OsfHgahyt-buY34dk0/view?usp=sharing

```
[ ]  %%bash
     pip  install  --upgrade  --no-cache-dir  gdown
     gdown  https://drive.google.com/uc?id=1tovCO2gsjesjJ8OsfHgahyt-buY34dk0
     unzip  aoi-dataset.zip
     rm  aoi-dataset.zip
```



Step 2: Import python libraries

AUAOI Ex2.

```
[ ]  import os
     import glob
     import torch
     from torch import nn
     from torch.utils.data import Dataset, DataLoader
     from torchvision import datasets
     from torchvision.transforms import ToTensor
```

```
[ ]  print (torch.cuda.is_available())
```

```
[ ]  device_name=torch.cuda.get_device_name(0)
     print(f"Using GPU {device_name}")
```

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     %matplotlib inline
```

Step 3: read the training set


```
[ ]  import pandas as pd
     df_train = pd.read_csv("train.csv")
     print(df_train.shape)
```

```
[ ]  df_train.head()
```

```
[ ]  train_files = df_train.iloc[:,0].values
     train_labels = df_train.iloc[:,1].values
     print(train_labels[:10])
```

```
[6]  df_train.head()
```

	ID	Label
0	train_00000.png	0
1	train_00001.png	1
2	train_00002.png	1
3	train_00003.png	5
4	train_00004.png	5

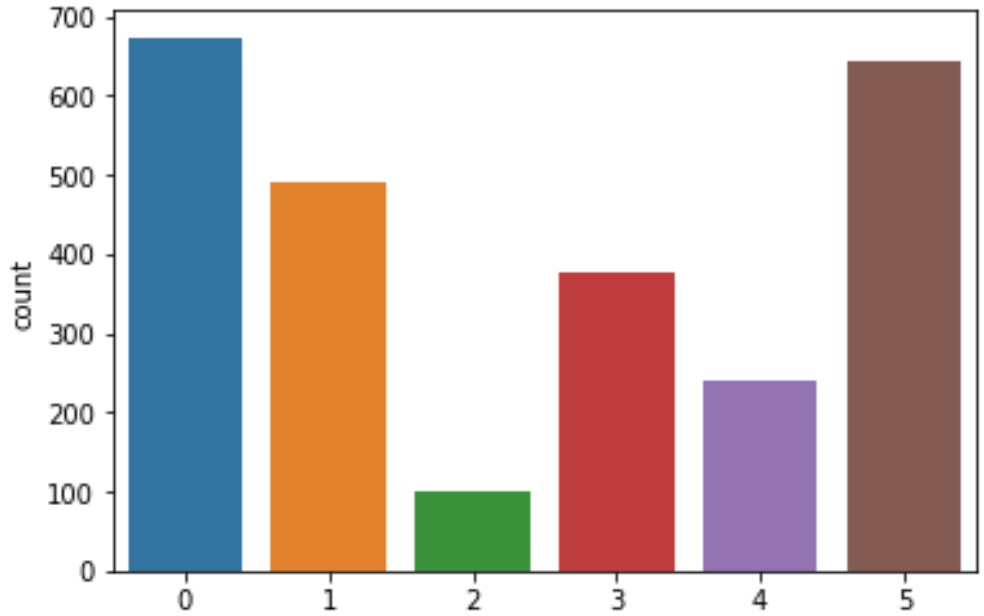
```
[7]  train_files = df_train.iloc[:,0].values
     train_labels = df_train.iloc[:,1].values
     print(train_labels[:10])
```

```
['0' '1' '1' '5' '5' '5' '3' '0' '3' '5']
```

Step 4: Show statistics of training images


```
import seaborn as sns
g = sns.countplot(train_labels)
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_tes
  import pandas.util.testing as tm
```



Step 5: Choose one of CNN models

EfficientNet B0 to B7

Model-EfficientNet

https://pytorch.org/hub/nvidia_deeplearningexamples_efficientnet/

Base model	resolution	Base model	resolution
EfficientNetB0	224	EfficientNetB4	380
EfficientNetB1	240	EfficientNetB5	456
EfficientNetB2	260	EfficientNetB6	528
EfficientNetB3	300	EfficientNetB7	600

```
[ ]  num_classes=6
```

```
[ ]  modelfile  =  None
```

```
[ ]  import  torchvision.models  as  models
     model=models.efficientnet_b0(num_classes=num_classes)
     if  modelfile  !=  None:  model.load_state_dict(torch.load(modelfile))
     model.cuda()
```

Step 6: Instancing a dataloader

- Transforms
- CustomDataset
- dataloader

```
from torchvision import transforms
pretrained_size = 224
pretrained_means = [0.485, 0.456, 0.406]
pretrained_stds= [0.229, 0.224, 0.225]
train_transform = transforms.Compose([
        transforms.Resize(pretrained_size),
        transforms.RandomRotation(5),
        transforms.RandomHorizontalFlip(0.5),
        transforms.RandomCrop(pretrained_size, pa
        transforms.ToTensor(),
        transforms.Normalize(mean = pretrained_m
])
```

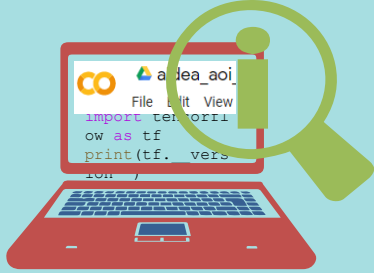
```
from PIL import Image
class CustomDataset(torch.utils.data.Dataset):
        def __init__(self, csv_path, images_folder, transform = None):
                self.df = pd.read_csv(csv_path)
                self.images_folder = images_folder
                self.transform = transform

        def __len__(self):
                return len(self.df)
        def __getitem__(self, index):
                filename = self.df.iloc[index]['ID']
                label = self.df.iloc[index]['Label']
                image = Image.open(os.path.join(self.images_folder, filename))
                if self.transform is not None:
                        image = self.transform(image)
                return image, label
```

Step 7: Set up a train dataloader with a custom dataset


```
batches = 24
imgdir= "train_images"
csvfile = "train.csv"
```

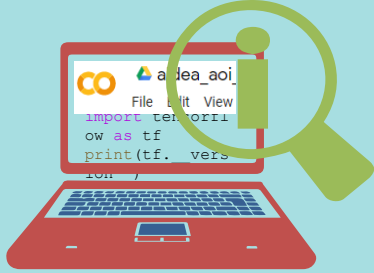
```
[ ] train_dataset = CustomDataset(csvfile, imgdir, train_transform)
    train_dataloader = DataLoader(train_dataset,batch_size=batches, shuffle=True)
    print(f"Total images={len(train_dataset)}")
```

Step 8: total_batch




```
total_batch=len(train_dataset)//batches  +  1
print(total_batch)
```

Step 9: Train model



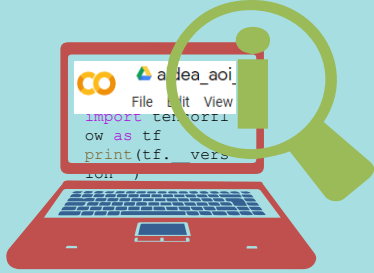
AUAOI Ex2.

```
loss = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

```
num_epochs = 10
```

```
for epoch in range(num_epochs):
    for i, (batch_images, batch_labels) in enumerate(train_dataloader):
        # Zero your gradients for every batch!
        optimizer.zero_grad()
        inputs = batch_images.cuda()
        labels = batch_labels.cuda()
        # Make predictions for this batch
        outputs = model(inputs)

        # Compute the loss and its gradients
        cost = loss(outputs , labels)
        cost.backward()
        # Adjust learning weights
        optimizer.step()

        if (i+1) % 100 == 0:
            print(f'Epoch [{epoch+1}/{num_epochs}], Iter [{i+1}/{total_batch}]
```

Step 10: Save the trained model




```
filepath  =  "AOI-EnBO.pth"
torch.save(model.state_dict(),  filepath)
```

Step 11: Check training results


```
model.eval()
#  again  no  gradients  needed
with  torch.no_grad():
        total_batch  =  len(test_dataset)//batches
        for  i,  (batch_images,  batch_labels)  in  enumerate(test_dataloader):
            images  =  batch_images.cuda()
            labels  =  batch_labels.cuda()
            outputs  =  model(images)
            _,  predictions  =  torch.max(outputs,  1)
            train_predictions[i*batches:(i+1)*batches]  =  predictions.cpu()
            if  (i+1)  %  10  ==  0:
                    print(f'Iter  [{i+1}/{total_batch}]')
```

```
train_predictions=train_predictions.astype(int)
```

Step 12: Analyze training results


```
from sklearn.metrics import confusion_matrix
confusion=confusion_matrix(train_labels, train_predictions)
print(confusion)
```

```
train_num = 2528
overkill= []
underkill = []
for i,(label, prediction) in enumerate(zip(train_labels, train_predictions)):
    if label == 0 and prediction !=0:
        overkill.append(i)
    if label != 0 and prediction ==0:
        underkill.append(i)
print('# of overkill= {}; # of underkill= {} '.format(len(overkill), len(underkill)))
```

Thanks!

Q&A