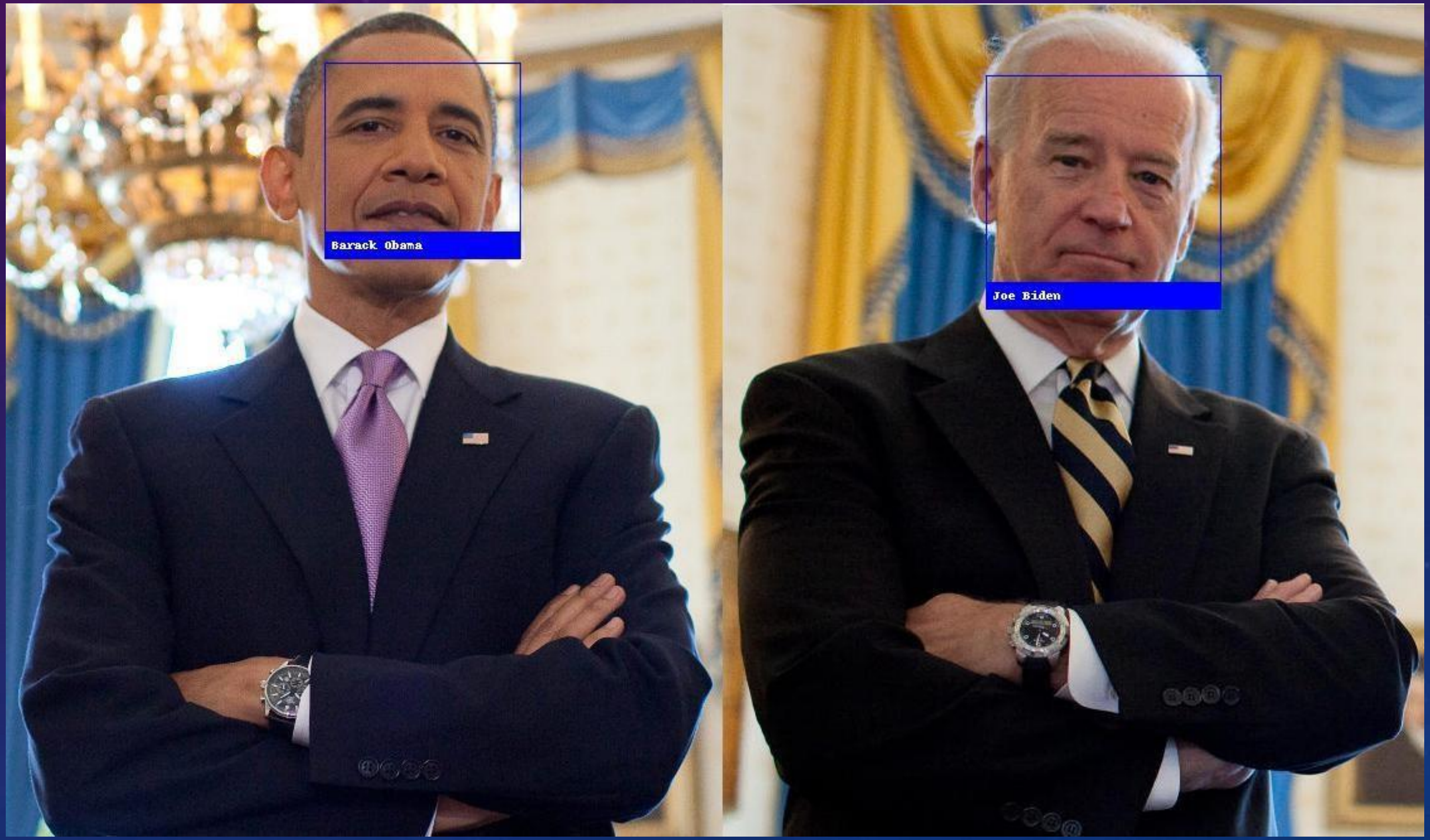
Digital Surveillance Systems and Application

Coding manual

Jison G.S. Hsu
Artificial Vision Laboratory
Taiwan Tech

Example 7.1 : Face Recognition

- Please use the dlib models on Colab to recognize the faces in image and show the result with detection boxes as shown below



Sol 7.1 : Face Recognition (1/10) – Face Verification

- Face verification is a 1: 1 comparison.
- The identity verification mode is essentially a process by which the computer quickly compares the current face with the portrait database and determines whether it matches.

```
!pip install face_recognition
!pip install pillow
!pip install numpy
```

```
#face detection part
import face_recognition
image = face_recognition.load_image_file('./two_people.jpg') #load image
face_locations = face_recognition.face_locations(image) #detect face

#Array of coords of each face
print(face_locations)
print(f'There are {len(face_locations)} people in this image')
```

Test Pair

Threshold:0.5

True

False

```
#face recognition part
import face_recognition
obama_1=face_recognition.load_image_file('./obama.jpg')
#feature extractor
obama_1_encoding = face_recognition.face_encodings(obama_1)[0]

unknown_im=face_recognition.load_image_file('./biden.jpg')
unknown_im_encoding = face_recognition.face_encodings(unknown_im)[0]

# feature compare
results = face_recognition.compare_faces([obama_1_encoding],unknown_im_encoding)
if results[0]:
  print('This is obama')
else:
  print('This is not obama')
```

Sol 7.1 : Face Recognition (2/10)

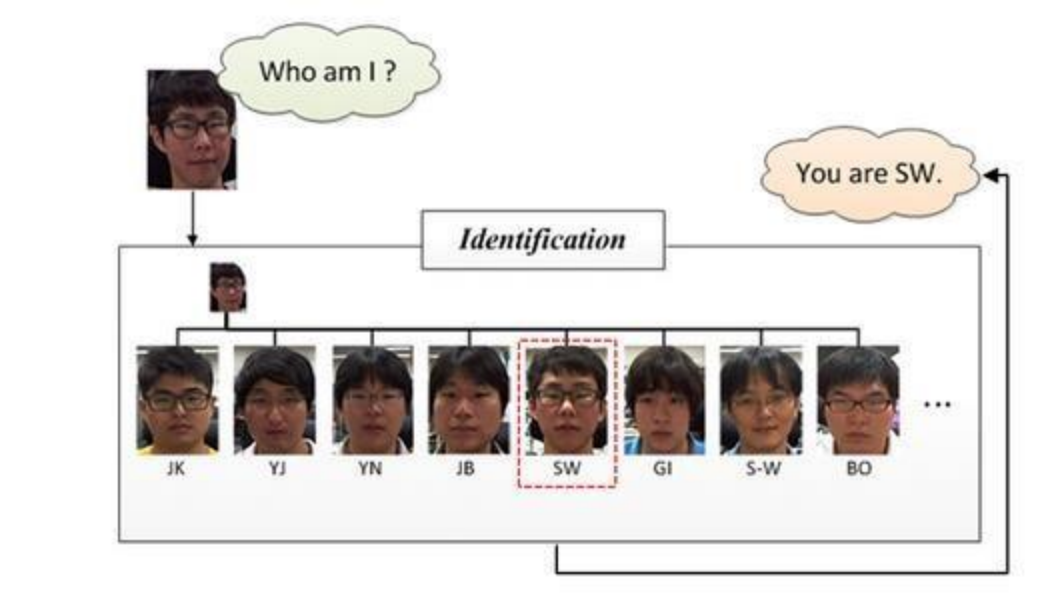
```python
# face detection and show the part of face
from PIL import Image
import face_recognition
image = face_recognition.load_image_file('./two_people.jpg')
face_locations = face_recognition.face_locations(image)

for face_location in face_locations:
  top, right, bottom, left = face_location

  face_image = image[top:bottom, left:right]
  pil_image = Image.fromarray(face_image)
  display(pil_image)
  pil_image.save(f'{top}.jpg')
```

Sol 7.1 : Face Recognition (3/10) – Face Identification

- Face identification is as shown in the figure below. It will answer "Who am I?“
- Compared with pair matching used in face verification, it uses classification more in the recognition stage.




```python
# This is an example of running face recognition on a single image
# and drawing a box around each person that was identified.
import face_recognition
from PIL import Image, ImageDraw
import numpy as np



# Load a sample picture and get the feature.
obama_image = face_recognition.load_image_file("obama.jpg")
obama_face_encoding = face_recognition.face_encodings(obama_image)[0]

# Load a second sample picture and get the feature.
biden_image = face_recognition.load_image_file("biden.jpg")
biden_face_encoding = face_recognition.face_encodings(biden_image)[0]

# Create arrays of known face feature and their names
known_face_encodings = [
    obama_face_encoding,
    biden_face_encoding
]
known_face_names = [
    "Barack Obama",
    "Joe Biden"
]
```

Sol 7.1 : Face Recognition (4/10) – Face Identification

```python
# Load an image with an unknown face
unknown_image = face_recognition.load_image_file("obama_group.jpg")

# Find all the faces and face features in the unknown image
face_locations = face_recognition.face_locations(unknown_image)
face_encodings = face_recognition.face_encodings(unknown_image, face_locations)

# Convert the image to a PIL-format image so that we can draw on top of it with the Pillow library
pil_image = Image.fromarray(unknown_image)
# Create a Pillow ImageDraw Draw instance to draw with
draw = ImageDraw.Draw(pil_image)
```

```python
# Loop through each face found in the unknown image
for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):
    # See if the face is a match for the known face(s)
    matches = face_recognition.compare_faces(known_face_encodings, face_encoding)

    name = "Unknown"

    # If a match was found in known_face_encodings, just use the first one.
    # if True in matches:
    #     first_match_index = matches.index(True)
    #     name = known_face_names[first_match_index]

    # Or instead, use the known face with the smallest distance to the new face
    face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
    best_match_index = np.argmin(face_distances)
    if matches[best_match_index]:
        name = known_face_names[best_match_index]
```

Sol 7.1 : Face Recognition (5/10) – Face Identification

```python
    # Draw a box around the face using the Pillow module
    draw.rectangle(((left, top), (right, bottom)), outline=(0, 0, 255))

    # Draw a label with a name below the face
    text_width, text_height = draw.textsize(name)
    draw.rectangle(((left, bottom - text_height - 10), (right, bottom)), fill=(0, 0, 255), outline=(0, 0, 255))
    draw.text((left + 6, bottom - text_height - 5), name, fill=(255, 255, 255, 255))


# Remove the drawing library from memory as per the Pillow docs
del draw

# Display the resulting image
display(pil_image)

# You can also save a copy of the new image to disk if you want by uncommenting this line
pil_image.save("image_with_boxes.jpg")
```

Threshold:0.7

Sol 7.1 : Face Recognition (6/10) – Face Identification

Threshold:0.5

Threshold:0.35

Sol 7.1 : Face Recognition (7/10) – Function Explanation

```
def face_locations(img, number_of_times_to_upsample=1, model="hog"):
    """
    Returns an array of bounding boxes of human faces in a image
    :param img: An image (as a numpy array)
    :param number_of_times_to_upsample: How many times to upsample the image looking for faces. Higher
numbers find smaller faces.
    :param model: Which face detection model to use. "hog" is less accurate but faster on CPUs. "cnn" is a more
accurate deep-learning model which is GPU/CUDA accelerated (if available). The default is "hog".
    :return: A list of tuples of found face locations in css (top, right, bottom, left) order
    """
    if model == "cnn":
        return [_trim_css_to_bounds(_rect_to_css(face.rect), img.shape) for face in _raw_face_locations(img,
number_of_times_to_upsample, "cnn")]
    else:
        return [_trim_css_to_bounds(_rect_to_css(face), img.shape) for face in _raw_face_locations(img,
number_of_times_to_upsample, model)]
```

Sol 7.1 : Face Recognition (8/10) – Function Explanation

```
def _raw_face_landmarks(face_image, face_locations=None, model="large"):
    if face_locations is None:
        face_locations = _raw_face_locations(face_image)
    else:
        face_locations = [_css_to_rect(face_location) for face_location in face_locations]

    pose_predictor = pose_predictor_68_point #dlib.shape_predictor(predictor_68_point_model)

    if model == "small":
        pose_predictor = pose_predictor_5_point #dlib.shape_predictor(predictor_5_point_model)

    return [pose_predictor(face_image, face_location) for face_location in face_locations]
```

Sol 7.1 : Face Recognition (9/10) – Function Explanation

```
def face_encodings(face_image, known_face_locations=None, num_jitters=1, model="small"):
    """
    Given an image, return the 128-dimension face encoding for each face in the image.
    :param face_image: The image that contains one or more faces
    :param known_face_locations: Optional - the bounding boxes of each face if you already know them.
    :param num_jitters: How many times to re-sample the face when calculating encoding. Higher is more accurate,
but slower (i.e. 100 is 100x slower)
    :param model: Optional - which model to use. "large" (default) or "small" which only returns 5 points but is faster.
    :return: A list of 128-dimensional face encodings (one for each face in the image)
    """
    raw_landmarks = _raw_face_landmarks(face_image, known_face_locations, model)
    return [np.array(face_encoder.compute_face_descriptor(face_image, raw_landmark_set, num_jitters)) for
raw_landmark_set in raw_landmarks]
#face_encoder = dlib.face_recognition_model_v1(face_recognition_model)
```

Sol 7.1 : Face Recognition (10/10) – Function Explanation

```
def face_distance(face_encodings, face_to_compare):
    """
    Given a list of face encodings, compare them to a known face encoding and get a euclidean distance
    for each comparison face. The distance tells you how similar the faces are.
    :param faces: List of face encodings to compare
    :param face_to_compare: A face encoding to compare against
    :return: A numpy ndarray with the distance for each face in the same order as the 'faces' array
    """
    if len(face_encodings) == 0:
        return np.empty((0))

    return np.linalg.norm(face_encodings - face_to_compare, axis=1)
```

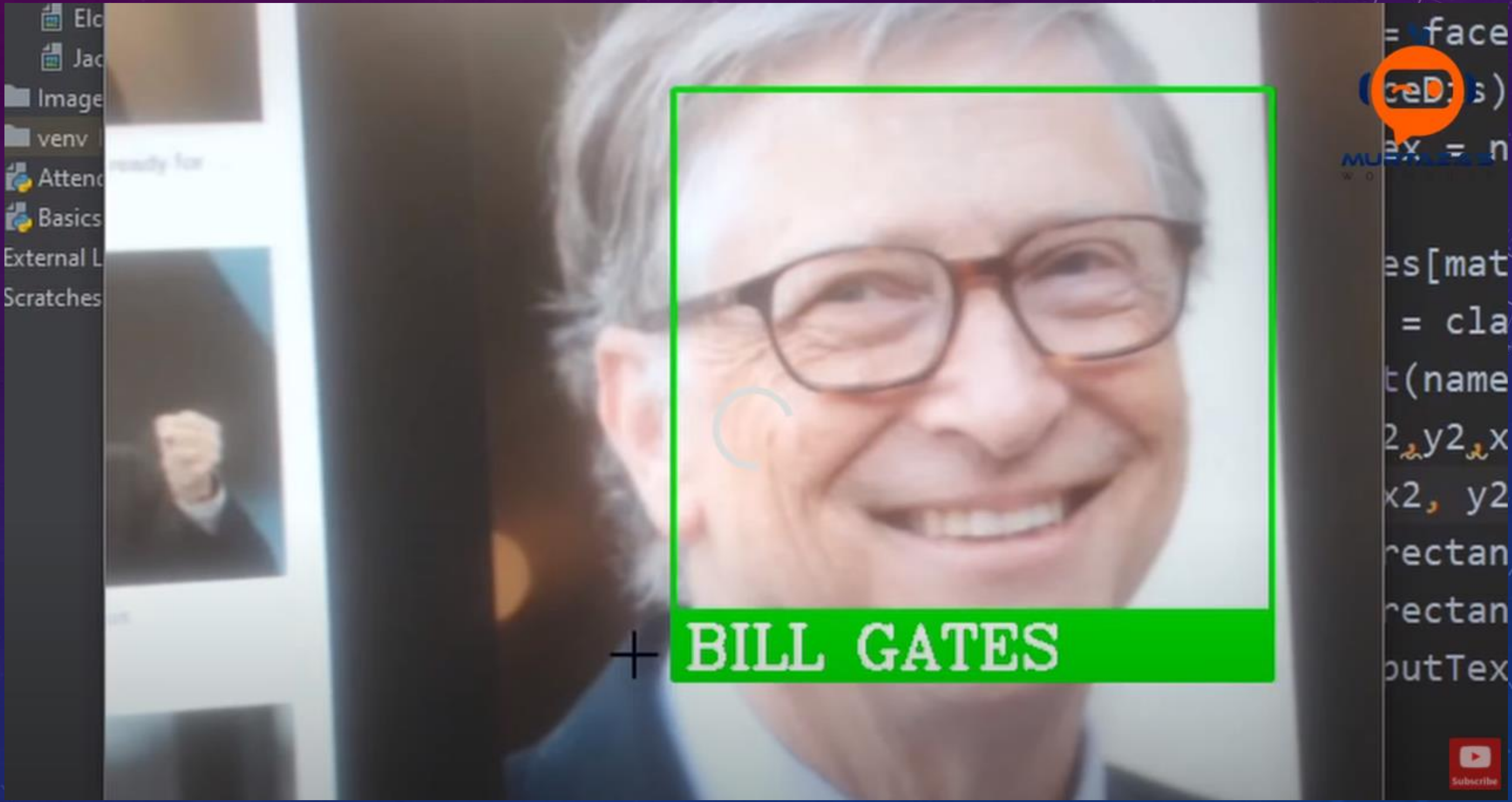
Exercise 7.1 : Face Recognition

Please use face_recognition.ipynb from Moodle to solve the following problems

1. Face identification : Choose 10 subjects and one img/subject from the web to make a gallery set, and make a test set of 10 images which show different poses, lightings and expressions, and contain 2~5 subjects in the gallery set. Use different thresholds to plot the FRR v.s. FAR
2. Face verification : Compare your own photos and the images of celebrities on the web, use different thresholds for the verification test and show the corresponding feature distances & results.

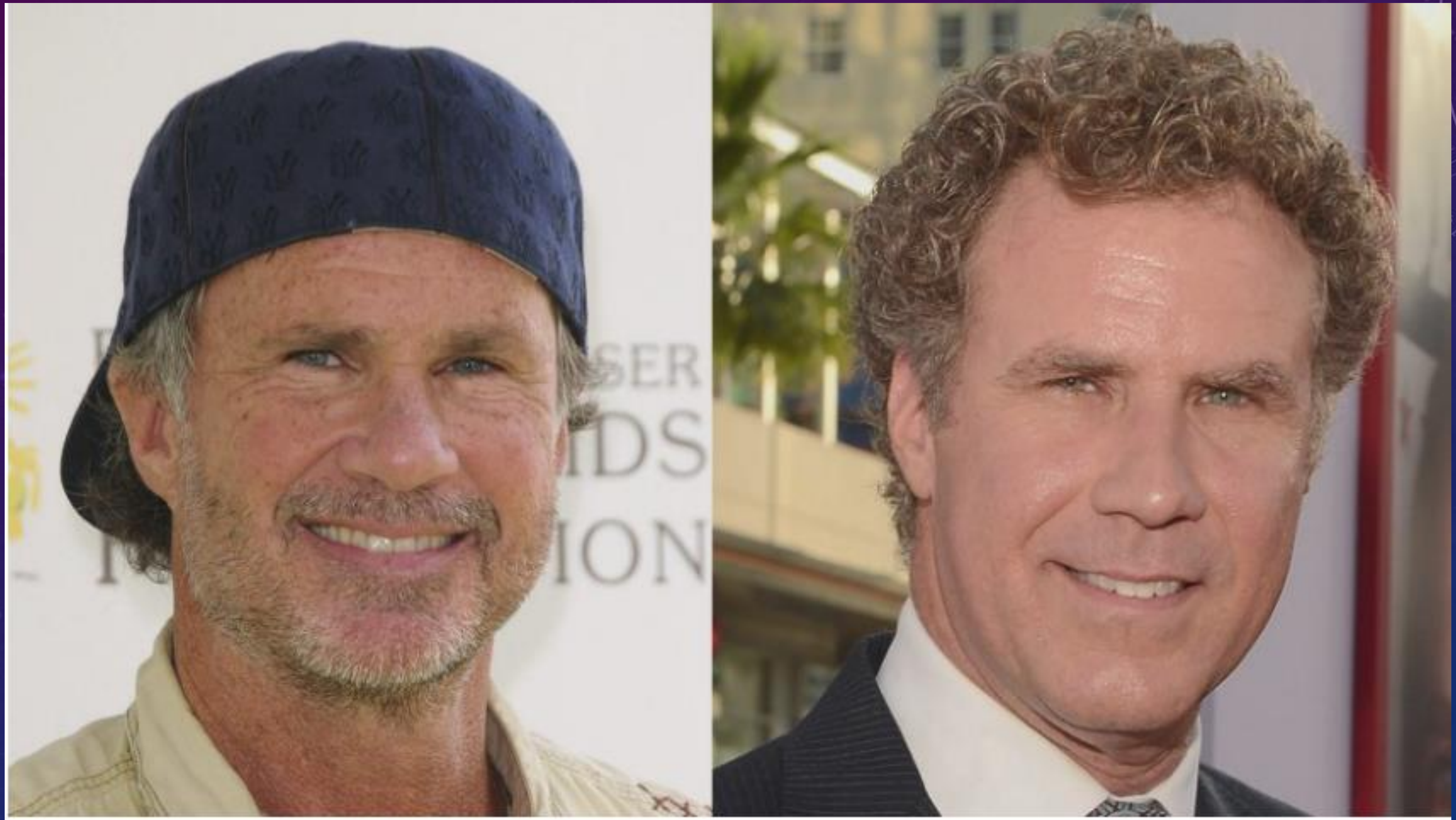
Please upload your code and results to Moodle

FACE RECOGNITION + ATTENDANCE PROJECT



https://youtu.be/sz25xxF_AVE [52:23]

Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning



One of these people is Will Farrell. The other is Chad Smith. I swear they are different people!

https://medium.com/@ageitgey/ machine-learning-is-fun-part- 4-modern-face-recognition- with-deep-learning- c3cffc121d78

Pixel-Level Recognition

https://youtu.be/QCtHGT68RIU [28:56]

LOSS FUNCTION IMPLEMENTATION

Example 7.2 : Loss function implementation

Prepare image for your task

⇩

Create Training / Validation / Testing List

⇩

Modify the model

⇩

Change different loss function

⇩

Test

Installation Process

- Download **loss_function.ipynb** from Moodle and open it
- Download **CASIA-DATA.zip** and unzip it to “./Database/”

```
%cd  /content/DSS_Loss_Function
!unzip  ./Database.zip
!wget  https://www.dropbox.com/s/tdkptvd36nvx6ox/CASIA-Data.zip?dl=0  -O  ./Database/CASIA-Data.zip
!unzip  ./Database/CASIA-Data.zip  -d  ./Database
```

- Images are picked from CASIA FACE database.
  - 20 subjects
  - Label: 0~20

Database: CASIA-WebFace

- Unconstrained environment
- 10,575 identities with 494,414 images
- Collected from Internet Movie Database (IMDB)
- Large training data in public

Create Training/Validation/Testing List

Find create_csv.py

Use command to run create.py

```
!python  create_csv.py
```

After execution, you can find“ train.csv”, “test.csv”and “ val.csv” under the “DataList” folder.

Create Training/Validation/Testing List

- Split training images in each subject for **Training, Validation** and **Test.**

```
import os
import csv
idx = 0
person = 0
with open(R'./DataList/train.csv','w', newline = '') as train:
    with open(R'.\DataList\val.csv','w', newline = '') as val:
        with open(R'.\DataList\test.csv','w', newline = '') as test:
            csv_writer = csv.writer(train)
            csv_writer.writerow(['image','id'])
            csv_writer1 = csv.writer(val)
            csv_writer1.writerow(['image','id'])
            csv_writer2 = csv.writer(test)
            csv_writer2.writerow(['image','id'])
            for root, dirs, filenames in  os.walk(r'.\Database\CASIA-Data'):
                for k in dirs:
                    for root1, dirs1, filenames1 in os.walk(r'.\Database\CASIA-
Data\{}'.format(k)):
                        for i in filenames1:
                            idx += 1
                            data_path = os.path.join(root1,i)
                            print(idx)
                            if len(filenames1)*0 < idx <= len(filenames1)*0.8:
                                csv_writer.writerow([data_path,person])
                            if len(filenames1)*0.8 < idx <= len(filenames1)*0.9:
                                csv_writer1.writerow([data_path,person])
                            if len(filenames1)*0.9 < idx <= len(filenames1)*1.0:
                                csv_writer2.writerow([data_path,person])
                    idx = 0
                    person += 1
```

CASIA-DATA path

80% for Training
10% for Validation
10% for Test

Create Training/Validation/Testing List

This is “train.csv”.

The format is the same as “test.csv”and, “val.csv”

	A	B
1	image	id
2	.\Database\CASIA-Data\0000045\001.bmp	0
3	.\Database\CASIA-Data\0000045\002.bmp	0
4	.\Database\CASIA-Data\0000045\003.bmp	0
5	.\Database\CASIA-Data\0000045\004.bmp	0
6	.\Database\CASIA-Data\0000045\005.bmp	0
7	.\Database\CASIA-Data\0000045\006.bmp	0
8	.\Database\CASIA-Data\0000045\007.bmp	0
9	.\Database\CASIA-Data\0000045\008.bmp	0
10	.\Database\CASIA-Data\0000045\009.bmp	0
11	.\Database\CASIA-Data\0000045\011.bmp	0
12	.\Database\CASIA-Data\0000045\012.bmp	0
13	.\Database\CASIA-Data\0000099\001.bmp	1
14	.\Database\CASIA-Data\0000099\002.bmp	1
15	.\Database\CASIA-Data\0000099\003.bmp	1
16	.\Database\CASIA-Data\0000099\004.bmp	1
17	.\Database\CASIA-Data\0000099\005.bmp	1

Images path

Images label

Now, open the “Main_Angular.py” Let’s check the code !

Model Parameter

All the argument value can be found here, and the explanation will be shown in help

```python
if __name__=="__main__":

    parser = argparse.ArgumentParser(description='DR_GAN')
    # learning & saving parameterss
    parser.add_argument('-train', action='store_true', default=True,
                        help='Generate pose modified image from given image')
    parser.add_argument('-lr', type=float, default=0.0001, help='initial learning rate [default: 0.0002]')
    parser.add_argument('-step-learning', action='store_true', default=False, help='enable lr step learning')
    parser.add_argument('-lr-decay', type=float, default=0.1, help='initial decay learning rate [default: 0.1]')
    parser.add_argument('-lr-step', type=int, default=35, help='Set Step to change lr by multiply lr-decay thru every lr-step epoch [default: 35]')
    parser.add_argument('-beta1', type=float, default=0.5, help='adam optimizer parameter [default: 0.5]')
    parser.add_argument('-beta2', type=float, default=0.999, help='adam optimizer parameter [default: 0.999]')
    parser.add_argument('-epochs', type=int, default=5, help='number of epochs for train [default: 1000]')
    parser.add_argument('-Train-Batch', type=int, default=8, help='batch size for training [default: 64]')
    parser.add_argument('-Val-Batch', type=int, default=32, help='batch size for training [default: 4]')
    parser.add_argument('-Test-Batch', type=int, default=32, help='batch size for training [default: 64]')
    parser.add_argument('-snapshot-dir', type=str, default='snapshot', help='where to save the snapshot while training')
    parser.add_argument('-save-freq', type=int, default=1, help='save learned model for every "-save-freq" epoch')
    parser.add_argument('-cuda', action='store_true', default=False, help='enable the gpu')
    parser.add_argument('-start-epoch', default=1, type=int, metavar='N', help='manual epoch number (useful on restarts)')
```

Model Parameter

```python
# data souce
parser.add_argument('-data-place', type=str, default=None, help='prepared data path to run program')
parser.add_argument('-output', type=str, default='Output', help='Output path for features')
parser.add_argument('-train-csv-file', type=str, default=None, help='csv file to load image for training')
parser.add_argument('-val-csv-file', type=str, default=None, help='csv file to load image for validation')
parser.add_argument('-test-csv-file', type=str, default=None, help='csv file to load image for test')
parser.add_argument('-Nd', type=int, default=10, help='initial Number of ID [default: 188]')
parser.add_argument('-Channel', type=int, default=3, help='initial Number of Channel [default: 3 (RGB Three Channel)]')
# option
parser.add_argument('-snapshot', type=str, default=None, help='filename of model snapshot(snapshot/{Single or Multiple}/{date}/{epoch}) [default: None]')
parser.add_argument('-test', action='store_true', default=None, help='Generate pose modified image from given image')
parser.add_argument('-resume', default='', type=str, metavar='PATH', help='path to latest checkpoint (default: none)')
parser.add_argument('-Angle-Loss', action='store_true', default=False, help='Use Angle Loss')
parser.add_argument('-pretrain', action='store_true', default=False)

args = parser.parse_args()
writer = SummaryWriter()
```

Whether use Angular Softmax Loss or not, the default setting is Softmax with cross entropy loss

Training Mode

Choose the training mode in argument value

```python
elif args.train:
    print("Parameters:")
    for attr, value in sorted(args.__dict__.items()):
        text = "\t{}={}\n".format(attr.upper(), value)
        print(text)
        with open('{}/Parameters.txt'.format(args.snapshot_dir), 'a') as f:
            f.write(text)

    if args.train_csv_file is None or args.val_csv_file is None:
        print(">>> Sorry, please set csv-file for your training/validation data")
        exit()

    else:
        Model = VGG16(args)
        print(Model)
        Train(Model, args)

```

Set VGG16 as Model and print out the architecture

VGG-16

- Check DSS_Loss_Function/model/VGG16_Model.py

```python
class VGG16(nn.Module):
    def __init__(self, args, init_weights=True):
        super(VGG16, self).__init__()
        self.features = []
        self.Nd = args.Nd
        self.Channel = args.Channel
        self.AngleLoss = args.Angle_Loss

        ConvBlock1 = [
            nn.Conv2d(self.Channel, 64, 3, 1, 1),  # conv1_1
            nn.BatchNorm2d(64),
            nn.ELU(),
            nn.Conv2d(64, 64, 3, 1, 1),  # conv1_2
            nn.BatchNorm2d(64),
            nn.ELU(),
            nn.MaxPool2d(2, stride=2),  # pool1
        ]
        ConvBlock2 = [
            nn.Conv2d(64, 128, 3, 1, 1),  # conv2_1
            nn.BatchNorm2d(128),
            nn.ELU(),
            nn.Conv2d(128, 128, 3, 1, 1),  # conv2_2
            nn.BatchNorm2d(128),
            nn.ELU(),
            nn.MaxPool2d(2, stride=2),  # pool2
        ]
```

```python
        ConvBlock3 = [
            nn.Conv2d(128, 256, 3, 1, 1),  # conv3_1
            nn.BatchNorm2d(256),
            nn.ELU(),
            nn.Conv2d(256, 256, 3, 1, 1),  # conv3_2
            nn.BatchNorm2d(256),
            nn.ELU(),
            nn.Conv2d(256, 256, 3, 1, 1),  # conv3_3
            nn.BatchNorm2d(256),
            nn.ELU(),
            nn.MaxPool2d(2, stride=2),  # pool3
        ]
        ConvBlock4 = [
            nn.Conv2d(256, 512, 3, 1, 1),  # conv4_1
            nn.BatchNorm2d(512),
            nn.ELU(),
            nn.Conv2d(512, 512, 3, 1, 1),  # conv4_2
            nn.BatchNorm2d(512),
            nn.ELU(),
            nn.Conv2d(512, 512, 3, 1, 1),  # conv4_3
            nn.BatchNorm2d(512),
            nn.ELU(),
            nn.MaxPool2d(2, stride=2),  # pool4
        ]
        ConvBlock5 = [
            nn.Conv2d(512, 512, 3, 1, 1),  # conv5_1
            nn.BatchNorm2d(512),
            nn.ELU(),
            nn.Conv2d(512, 512, 3, 1, 1),  # conv5_2
            nn.BatchNorm2d(512),
            nn.ELU(),
            nn.Conv2d(512, 512, 3, 1, 1),  # conv5_3
```

VGG-16

- Check DSS_Loss_Function/model/VGG16_Model.py

```
self.convLayers1 = nn.Sequential(*ConvBlock1)
self.convLayers2 = nn.Sequential(*ConvBlock2)
self.convLayers3 = nn.Sequential(*ConvBlock3)
self.convLayers4 = nn.Sequential(*ConvBlock4)
self.convLayers5 = nn.Sequential(*ConvBlock5)

self.FC6 = nn.Sequential(
    nn.Linear(2048, 4096),
    nn.ReLU(),
    nn.Dropout(0.5),
)
self.FC7 = nn.Sequential(
    nn.Linear(4096, 4096),
    nn.ReLU(),
    nn.Dropout(0.6),
)
if self.AngleLoss:
    self.FC8 = AngleLinear(4096, self.Nd)
else:
    self.FC8 = nn.Linear(4096, self.Nd)
```

Decide how does the data flow in structure

```
def forward(self, input, ExtractMode=False):

    x1 = self.convLayers1(input)
    x2 = self.convLayers2(x1)
    x3 = self.convLayers3(x2)
    x4 = self.convLayers4(x3)
    x = self.convLayers5(x4)

    x = x.view(np.shape(x)[0], -1) # np.shape(x)[0] -> batch
    x = self.FC6(x)
    x = self.FC7(x)
    self.features = x
    x = self.FC8(x)

    if ExtractMode:
        return self.features
    else:
        return x
```

Feature extraction

Introduction VGG-16

Double ConvBlock

Double ConvBlock

Triple ConvBlock

Triple ConvBlock

Triple ConvBlock

Triple FC Layer

```
VGG16(
  (convLayers1): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ELU(alpha=1.0)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (convLayers2): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ELU(alpha=1.0)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (convLayers3): Sequential(
    (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ELU(alpha=1.0)
    (6): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ELU(alpha=1.0)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (convLayers4): Sequential(
    (0): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ELU(alpha=1.0)
    (6): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ELU(alpha=1.0)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (convLayers5): Sequential(
    (0): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ELU(alpha=1.0)
    (6): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ELU(alpha=1.0)
    (9): AdaptiveAvgPool2d(output_size=(2, 2))
  )
  (FC6): Sequential(
    (0): Linear(in_features=2048, out_features=4096, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5)
  )
  (FC7): Sequential(
    (0): Linear(in_features=4096, out_features=4096, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.6)
  )
  (FC8): Linear(in_features=4096, out_features=10, bias=True)
)
```

Training Process

```
def Train(Model, args):

    Nd = args.Nd
    beta1_Adam = args.beta1
    beta2_Adam = args.beta2

    if args.cuda:
        Model.cuda()

    optimizer = optim.Adam(Model.parameters(), lr=args.lr, betas=(beta1_Adam, beta2_Adam))
    Model.train()
    steps = 0
    CUDNN.benchmark = True

    for epoch in range(args.start_epoch, args.epochs+1):

        if args.step_learning:
            adjust_learning_rate(optimizer, epoch, args)

        transformed_dataset = FaceIdPoseDataset(args.train_csv_file, transform=transforms.Compose(
            [transforms.Resize(32),
             transforms.RandomCrop(28),
             transforms.ToTensor()]))
        dataloader = DataLoader(transformed_dataset, batch_size=args.Train_Batch, shuffle=True)
```

Decide the way to optimize model (default：Adam)

Original CASIA input is 112x112, resize into 256x256 and random crop back to 224x224 as data augmentation.

Training Process

```python
for i, batch_data in enumerate(dataloader):
    Model.zero_grad()
    batch_image = torch.FloatTensor(batch_data[0].float())
    batch_id_label = batch_data[2]
    if args.cuda:
        batch_image, batch_id_label = batch_image.cuda(), batch_id_label.cuda()
    batch_image, batch_id_label = Variable(batch_image), Variable(batch_id_label)

    steps += 1

    Prediction = Model(batch_image)
    Loss = Model.ID_Loss(Prediction, batch_id_label)

    Loss.backward()
    optimizer.step()
    log_learning(epoch, steps, 'VGG16_Model', args.lr, Loss.data, args)
    writer.add_scalar('Train/Train_Loss', Loss, steps)
    # Validation_Process(Model, epoch, writer, args)
Validation_Process(Model, epoch, writer, args)
```

Send the image to Model, and output the prediction.

Loss function

Define the loss function, we already prepare two kinds of loss function for you guys

```
loss_criterion = nn.CrossEntropyLoss().cuda()
loss_criterion_Angular = AngleLoss().cuda()
```

Given the FC prediction and label, than we can calculate the loss

```
    def ID_Loss(self, predic, label):

        if self.AngleLoss:
            Loss = loss_criterion_Angular((predic[0][:, :self.Nd], predic[1][:, :self.Nd]), label)
        else:
            Loss = loss_criterion(predic[:, :self.Nd], label)


        return Loss
```

Train the Model

Direct to the file path and input the command in the Terminal：

(Please build pytorch-gpu with the following command

conda install pytorch==1.0.0 torchvision==0.2.1 cuda80 -c pytorch)

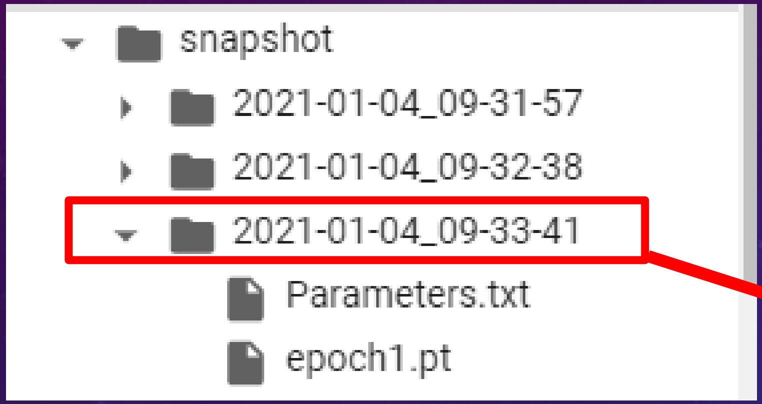
```
!python Main_Angular.py -cuda -train -data-place=./Database/CASIA-Data -train-csv-file=/content/DSS_Loss_Function/DataList/train.csv -val-csv-file=/content/DSS_Loss_Function/DataList/val.csv -Nd=20 -Train-Batch=8
```

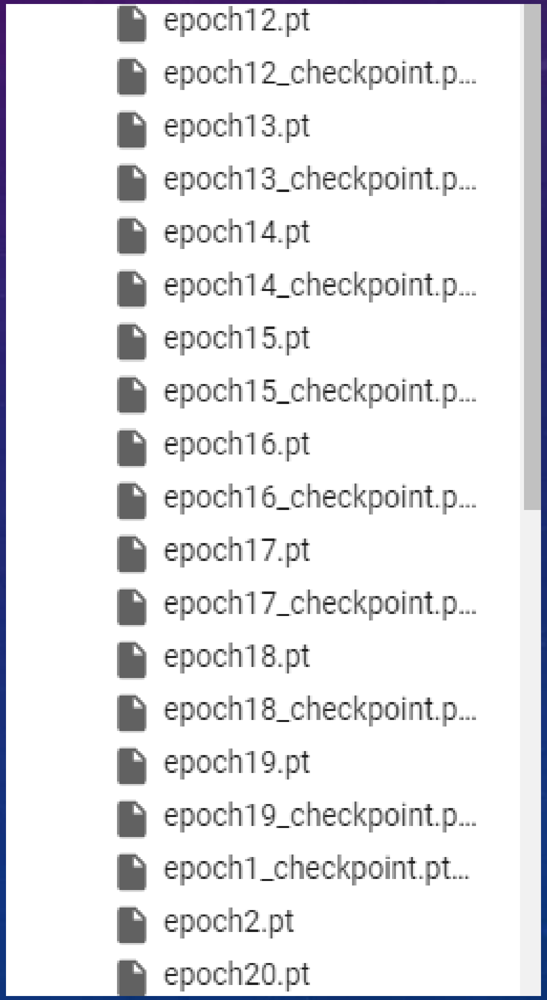
```
Fri, 22 Nov 2019 11:46:25 +0000 EPOCH : 1, step : 259, VGG16_Model_Lr: 0.0001, VGG16_Model, 3.203601598739624
Fri, 22 Nov 2019 11:46:25 +0000 EPOCH : 1, step : 260, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.9001379013061523
Fri, 22 Nov 2019 11:46:25 +0000 EPOCH : 1, step : 261, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.620979070663452
Fri, 22 Nov 2019 11:46:25 +0000 EPOCH : 1, step : 262, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.2005977630615234
Fri, 22 Nov 2019 11:46:25 +0000 EPOCH : 1, step : 263, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.6367361545562744
Fri, 22 Nov 2019 11:46:25 +0000 EPOCH : 1, step : 264, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.5872464179992676
Fri, 22 Nov 2019 11:46:25 +0000 EPOCH : 1, step : 265, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.5509464740753174
Fri, 22 Nov 2019 11:46:25 +0000 EPOCH : 1, step : 266, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.907942533493042
Fri, 22 Nov 2019 11:46:26 +0000 EPOCH : 1, step : 267, VGG16_Model_Lr: 0.0001, VGG16_Model, 3.0043704509735107
Fri, 22 Nov 2019 11:46:26 +0000 EPOCH : 1, step : 268, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.466834545135498
Fri, 22 Nov 2019 11:46:26 +0000 EPOCH : 1, step : 269, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.3059823513031006
Fri, 22 Nov 2019 11:46:26 +0000 EPOCH : 1, step : 270, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.745974063873291
Fri, 22 Nov 2019 11:46:26 +0000 EPOCH : 1, step : 271, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.6620829105377197
Fri, 22 Nov 2019 11:46:26 +0000 EPOCH : 1, step : 272, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.74006986618042
Fri, 22 Nov 2019 11:46:26 +0000 EPOCH : 1, step : 273, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.995265483856201
Fri, 22 Nov 2019 11:46:26 +0000 EPOCH : 1, step : 274, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.3112964630126953
Fri, 22 Nov 2019 11:46:27 +0000 EPOCH : 1, step : 275, VGG16_Model_Lr: 0.0001, VGG16_Model, 2.737488031387329
Fri, 22 Nov 2019 11:46:28 +0000 EPOCH : 1, step : 276, VGG16_Model_Lr: 0.0001, VGG16_Model, 1.990052342414856
Start Validating...
>>> epoch: '1'
>>> ID_Precision: '0.14236111111111111'
>>> Validation_Loss: '2.692030191421509'
```

Training process look like

Model saved

The model will save every epoch, and save in VGG16/snapshot.





You will need this model in Test phase

Test Phase

Type the command below to extract the feature from the model

```
!python Main_Angular.py -cuda -test -Nd=20 -test-csv-file=/content/DSS_Loss_Function/DataList/test.csv -snapshot=./snapshot/2021-01-04_09-33-41/epoch20 -output=./Output
```

Change to your model path

```
>>> Loading model from [./snapshot/2019-11-22_11-45-44/epoch8]...
Finish Processing 32 images...
Finish Processing 64 images...
Finish Processing 96 images...
Finish Processing 128 images...
Finish Processing 160 images...
Finish Processing 192 images...
Finish Processing 224 images...
Finish Processing 256 images...
Finish Processing 286 images...
```

Test Phase

Open the “Feature_Compare.py” in VGG16

```
import os
Import sys
import numpy as np
import scipy.io as sio
from scipy.spatial.distance import cdist
from numpy.linalg import norm
import csv
def read_feature(path):
   elements1 = []
   with open(path) as file:
     for line in file:
        line = line.strip().split()
        elements1.append(line)
  feature = np.array(elements1)
  feature = feature[:,0].astype(np.float64)
  return feature


Feature_path = './Output/snapshot/2019-11-22_11-45-44/epoch8/Feature'
Feature_dir = os.listdir(Feature_path)
Feature_dir.sort(key=lambda x:int(x))
```

Change to your own feature path

In Test phase, we extract the feature vector,
(not the FC output layer)
and compare the distance between each sample.

We choose the first sample in each class as the gallery image, and compare with all probe (test) sample to check the minimize distance.

Test Phase

Open the “Feature_Compare.py” in VGG16

```
## Choose first sample as gallery
Gallery=[]
for ii in range(len(Feature_dir)):
    fea_file = os.listdir(os.path.join(Feature_path,Feature_dir[ii]))
    fea = read_feature(os.path.join(Feature_path,Feature_dir[ii],fea_file[0]))
    Gallery.append(fea.T)
Gallery = np.array(Gallery)
Acc = 0
counter = 0


## Compare distance
for ii in range(len(Feature_dir)):
    fea_file = os.listdir(os.path.join(Feature_path, Feature_dir[ii]))
    for jj in range(len(fea_file)):
        Probe = read_feature(os.path.join(Feature_path,Feature_dir[ii],fea_file[jj]))
        distance = cdist(Gallery, Probe.reshape(-1,1).T , 'euclidean')
        value = distance.min()
        position = np.where(distance == value)
        if position[0] == (ii):
            Acc = Acc+1
        counter = counter +1
accuracy = Acc/counter
print('The accuracy = 
{}%\n'.format(accuracy*100))
```

Pairwise distance between two sets of observation

Reference Performance

```
(C:\Users\Micky\Anaconda3\envs\pytorch-cpu) F:\DSS_2019\VGG_NEW\VGG16>python Feature_Compare.py
The accuracy = 28.32167832167832%
```

Exercise 7.2 : Loss function implementation

Please change the loss function to “Angular Softmax Loss” with the same setting you have done in “Sample 7.2”, and use feature comparison to calculate the performance.

1. Compare the performance between difference loss function (Angular Softmax/ Cross entropy with softmax function)
2. Try to convert the pairwise distance from “Euclidean” to “Cosine”

Write down your observation and upload the report in MS Word or PDF to Moodle

Exercise 7.3 : Age classification

Please change the loss function to “Angular Softmax Loss” with the same setting you have done in **Example 7.2**, and use feature comparison to calculate the performance.

1. Use an age database “age.zip” from Colab that divides age into four intervals for training an age classifier and split the dataset with 70% for training, 10% for validation, 20% for testing.
2. Compare the performance between difference loss functions (Angular Softmax/ Cross Entropy with Softmax function)
3. Try to convert the pairwise distance from “Euclidean” to “Cosine”

Hint : The parameter “Nd” depends on the number of categories you have

Elaborate your solution and upload it to Moodle. You are encouraged to use the data and define your own version of additional experiments, and merge them to the report.

Exercise 7.4 : Object classification

Please change the loss function to "Angular Softmax Loss" with the same setting you have done in **Example 7.2**, and use feature comparison to calculate the performance.

1. Use an object database "cifar10.zip" given from Colab that divides object into ten intervals for training an object classifier and split the dataset with 70% for training, 10% for validation, 20% for testing.
2. Compare the performance between difference loss functions (Angular Softmax/ Cross Entropy with Softmax function)
3. Try to convert the pairwise distance from "Euclidean" to "Cosine"

Hint : The parameter "Nd" depends on the number of categories you have

Elaborate your solution and upload it to Moodle. You are encouraged to use the data and define your own version of additional experiments, and merge them to the report.