Assignment 2:

ML & DL Basics

Computer Vision

National Taiwan University

Fall 2019

TA : 黃柏翔、劉彥廷

TA

- 劉彥廷
  - TA Time : Tue. & Thu. 15:30 - 16:30
  - Lab : 博理527
  - E-mail : R06942114@ntu.edu.tw

- 黃柏翔
  - TA Time : Mon. & Tue. 15:30 - 16:30
  - Lab : 博理527
  - E-mail : R06942138@ntu.edu.tw

Part 1

ROC Curve

ROC Curve

- The Receiver Operating Characteristic (ROC) curve describes the capability of a binary classifier as its discrimination threshold varies.

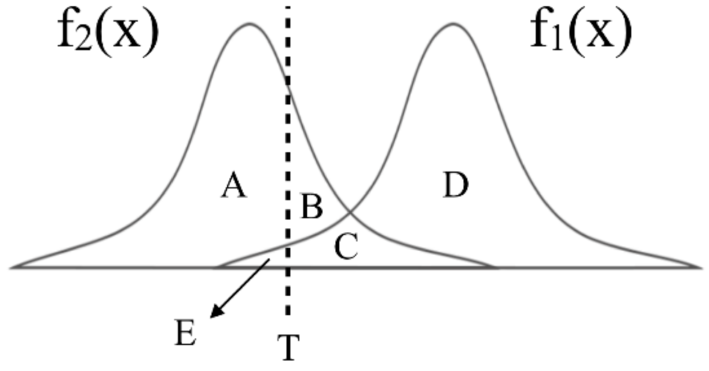
- Your answers should be as specific as possible. TAs have the right to determine the points you will receive based on the completeness of your answers.

Problem 1(a)

Assume X is a continuous random variable that denotes the estimated probability of a binary classifier. The instance is classified as *positive* if $X > T$ and *negative* otherwise.

When the instance is *positive*, X follows a PDF $f_1(x)$. When the instance is *negative*, X follows a PDF $f_2(x)$.

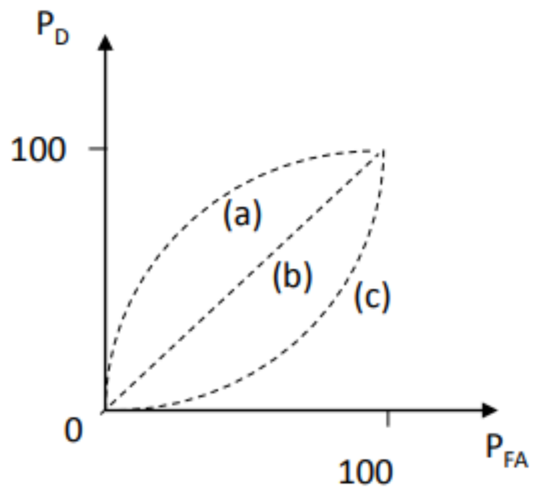
Please specify which regions (A ~ E) represent the cases of *False Positive* and *False Negative*, respectively. Clearly explain why.



Problem 1(b)

There are three ROC curves in the plot below. Please specify which ROC curves are considered to have reasonable discriminating ability, and which are not.

Also, please describe that under what circumstances will the ROC curve fall on curve (b)?



Problem 2

- Given a convolutional layer which contains
  n kernels with size $k * k * n_{in}$
  padding size (p, p),
  stride size (s, s).
  With input feature size $W * W * n_{in}$, calculate the size of output feature $W_{out} * W_{out} * n_{out}$.
  - $W_{out} = ?$, $n_{out} = ?$

- $n_{in} = 3, k = 5, p = 1, s = 2, n = 256, W = 64$, calculate the total number of parameters in the convolutional layer

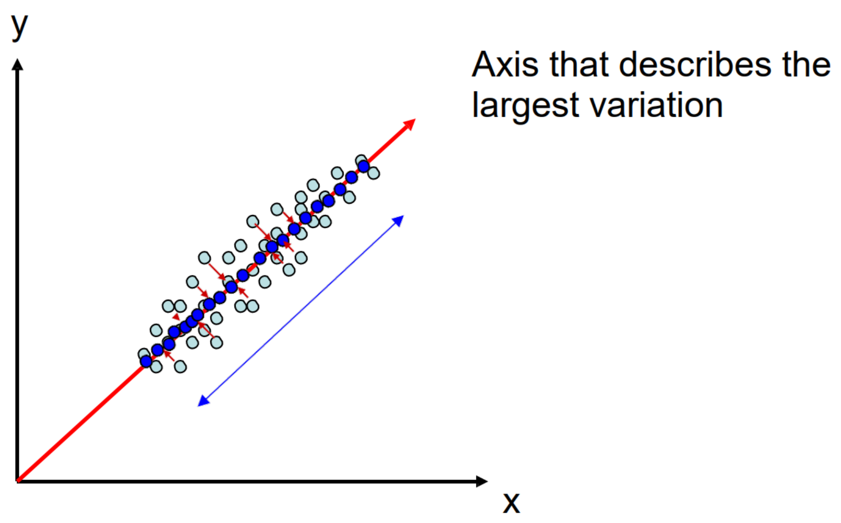
Part 2

PCA & KNN & K-means

Principal Component Analysis (PCA)

- PCA is an unsupervised dimension reduction technique.
  - In this part, we will practice PCA with some face images.

- We want to find projections of data (i.e., direction vectors that we can project the data onto) that describe the largest variation.



PCA – Eigenfaces

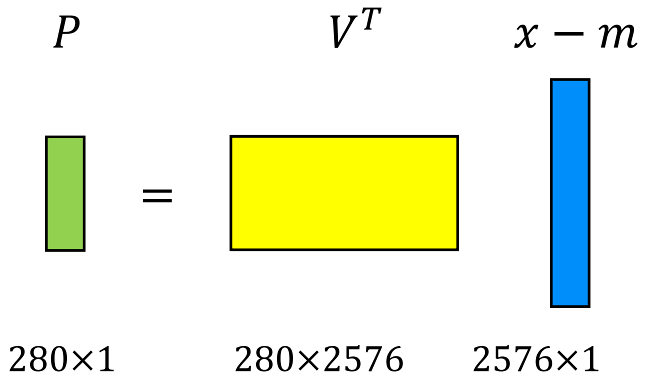
- $\Sigma v = \lambda v$
  - $\Sigma = \mathrm{E}[(X - \mathrm{E}[X])(X - \mathrm{E}[X])^T]$, covariance matrix
  - $X = [\ x^1\ \ x^2\ \ x^3 \ldots\ ]$

- The orthonormal eigenvectors of Σ are also known as eigenfaces when the data represents faces.

PCA – Image Reconstruction

- $V = \begin{bmatrix} (v_1)^T \\ (v_2)^T \\ \vdots \end{bmatrix}$

- $P = V^T(x - \underset{\text{mean face}}{\underset{\downarrow}{m}})$



- $x = \sum_{i=1}^{N} p_i v_i + m = p_1 \begin{bmatrix} | \\ v_1 \\ | \end{bmatrix} + p_2 \begin{bmatrix} | \\ v_2 \\ | \end{bmatrix} + \cdots + p_n \begin{bmatrix} | \\ v_n \\ | \end{bmatrix} + m$

  $= Vp + m$

Dataset

1_1.png

1_6.png

8_1.png

8_6.png

- Description:
  - Images collected from Olivetti faces dataset
  - Image shape: 56 x 46 x 1 (height x width x channel)
  - 40 different subjects, with 10 images available for each subject (Note that image "$\boldsymbol{i\_j.png}$" refers to "$\boldsymbol{person_i\_image_j}$")
- Data partition
  - We partition the dataset into training and testing sets as follows:
    - Training set
      - The first 7 images of each subject, 280 images in total
    - Testing set
      - The last 3 images of each subject, 120 images in total

Problem 3(a) – PCA

- In this task, you need to implement PCA **from scratch**, which means you **cannot** call PCA function directly from existing packages.

1. Perform PCA on the training data. Plot the **mean face** and the **first five eigenfaces** and show them in the report.
2. Take $\boldsymbol{person_8\_image_6}$, and project it onto the above PCA eigenspace. Reconstruct this image using the first n = { 5, 50, 150, all } eigenfaces. For each n, compute the **mean square error (MSE)** between the reconstructed face image and the original $\boldsymbol{person_8\_image_6}$. Plot these reconstructed images with the corresponding MSE values in the report.
3. Reduce the dimension of the image in testing set to **dim = 100**. Use t-SNE to visualize the distribution of test images.
4. (bonus) Implement the Gram Matrix trick for PCA. Compare the two reconstruction images of $\boldsymbol{person_8\_image_6}$ from standard PCA/Gram Matrix process.

Problem 3(b) – k-NN

To apply the k-nearest neighbors (k-NN) classifier to recognize the testing set images, please determine the best k and n values by **3-fold cross-validation**.

For simplicity, the choices for such hyper-parameters are:

$$k = \{1, 3, 5\} \text{ and } n = \{3, 50, 100\}.$$

Please show the cross-validation results and explain your choice for (k, n). Also, show the recognition rate on the testing set using your hyper-parameter choice.

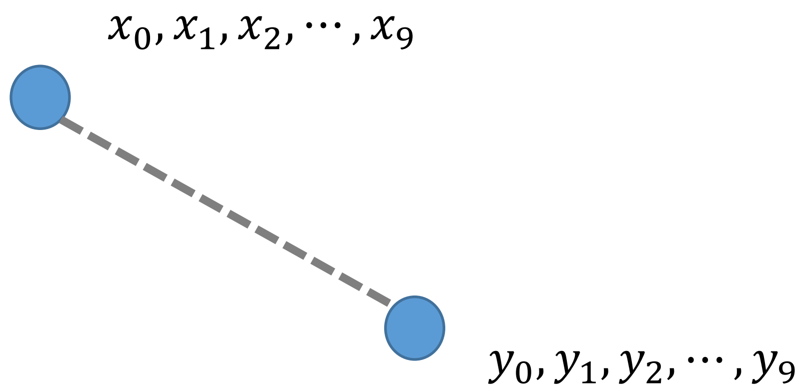
Problem 3(c) – k-means

1. **Reduce the dimension of the images in the first 10 class of training set to dim = 10 using PCA.** Implement K-means clustering to classify the these images from scratch, which means you cannot call K-means function directly from existing packages.
2. Please use weighted Euclidean distance to implement the k-means clustering. The weight of the 10 eigenfaces is

   w = [0.6, 0.4, 0.2, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]

3. Please visualize the **centroids** and features to 2D space with the first two eigenfaces for **each iteration** in k-means clustering.
4. Compare and discuss the final result of K-means and ground truth.

Distance



Euclidean Distance = $\sqrt{(x_0 - y_0)^2 + (x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_9 - y_9)^2}$

$w =$ [0.6, 0.4, 0.2, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]

Weighted Euclidean Distance =

$$\sqrt{w_0(x_0 - y_0)^2 + w_1(x_1 - y_1)^2 + w_2(x_2 - y_2)^2 + \cdots + w_9(x_9 - y_9)^2}$$

Part 4

Object Recognition using CNN

Problem 4(a)

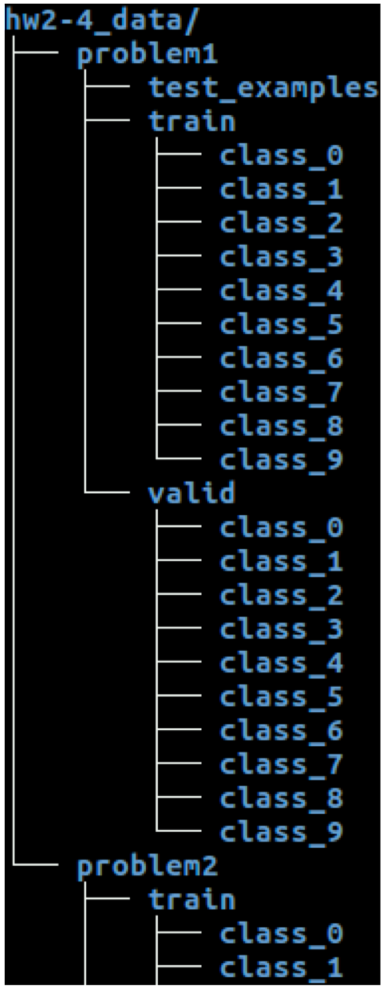
1. Build a “CNN model” and a “Fully-connected network” and train them on the given MNIST dataset. The prediction accuracy should pass the following baselines:
   - Validation accuracy: 98 % / 97%
   - Testing accuracy: 96 %
   - You may consider using a LeNet-5 as the CNN model.
2. Report the *architecture* and *number of parameters* of both networks.
3. Plot the learning curve (loss, accuracy) of the training process(train/validation). Compare the results of both model and explain the result.
4. **Use the code template provided by TA.**

Dataset

- Description:
  - Images collected from MNIST
  - Image shape: 28 x 28 x 1 (height x width x channel)

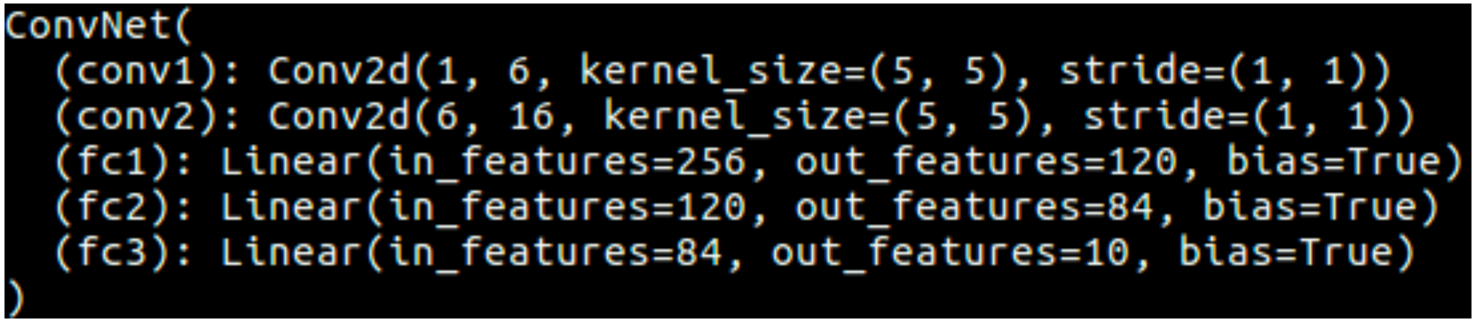
  - Training images:
    - 4000 images for each class, 40000 images in total
    - Images: 0000.png , … , 3999.png

  - Validation images:
    - 1000 images for each class, 10000 images in total
    - Images: 4000.png , … , 4999.png



```
hw2-4_data/
├── problem1
│   ├── test_examples
│   ├── train
│   │   ├── class_0
│   │   ├── class_1
│   │   ├── class_2
│   │   ├── class_3
│   │   ├── class_4
│   │   ├── class_5
│   │   ├── class_6
│   │   ├── class_7
│   │   ├── class_8
│   │   └── class_9
│   └── valid
│       ├── class_0
│       ├── class_1
│       ├── class_2
│       ├── class_3
│       ├── class_4
│       ├── class_5
│       ├── class_6
│       ├── class_7
│       ├── class_8
│       └── class_9
└── problem2
    ├── train
    │   ├── class_0
    │   ├── class_1
```

Example

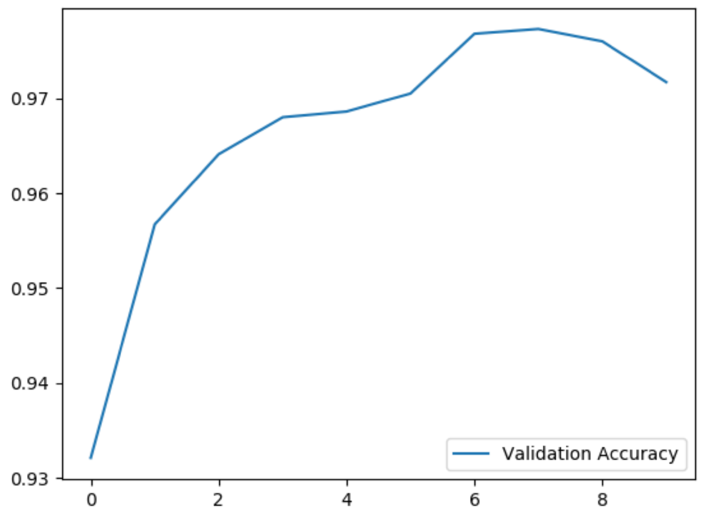


```
ConvNet(
  (conv1): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1))
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=256, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
```

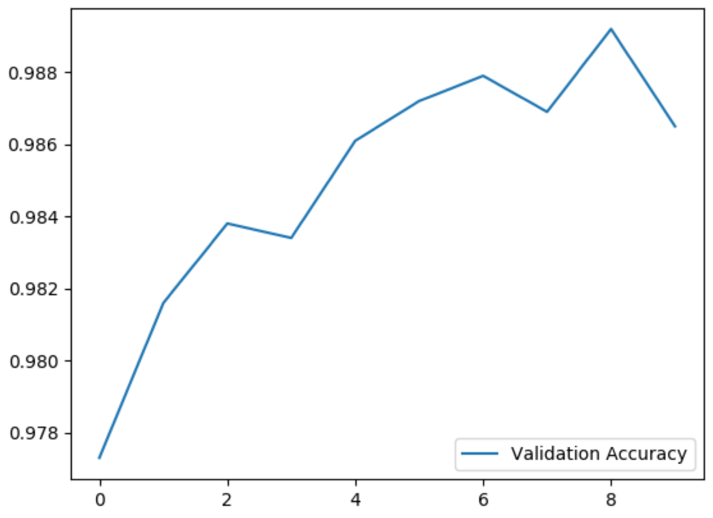
Parameters: 44426

Example

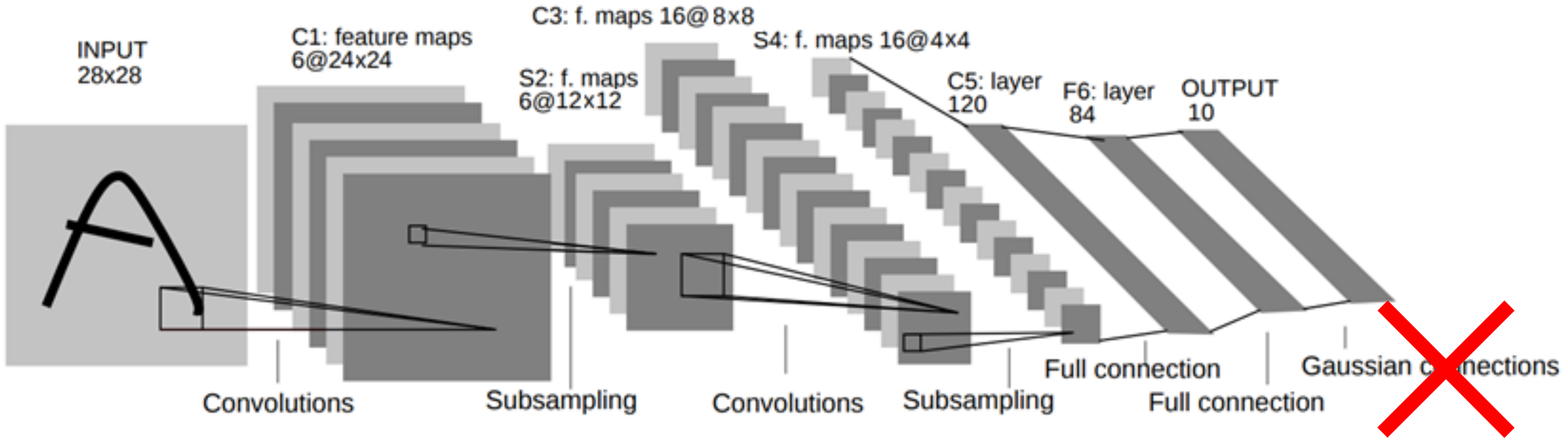
Fully Connected Network



Convolutional Network



LeNet-5



softmax

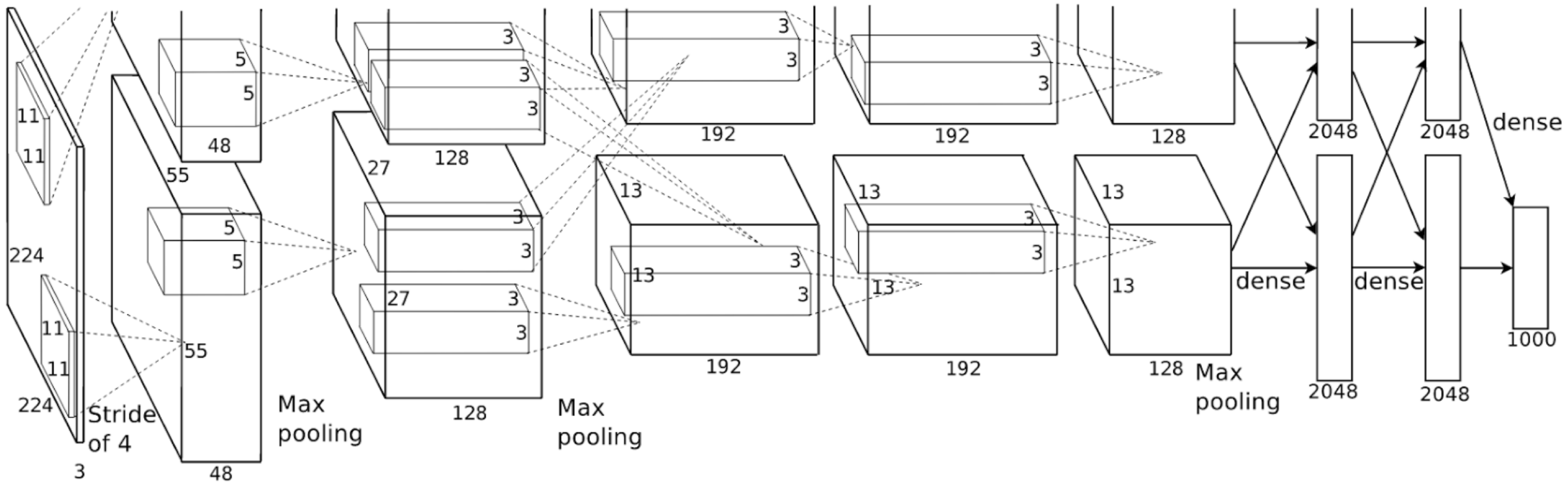
Yann LeCun, Leon Bottu, Yoshua Bengio, Patrick Haffner. [Gradient-Based Learning Applied to Document Recognition]

Problem 4(b)

1. Extract feature using ***pytorch pretrained alexnet*** and train a ***KNN classifier*** to perform human face recognition on the given dataset (you can use PCA to reduce dimension before training KNN). (TA baseline accuracy: 35%)
2. Build your own model and train it on the given dataset to surpass the accuracy of previous stage.
3. Plot the learning curve of your training process (training/validation loss/accuracy) of stage (2), explain the result you observed.
4. **Pick the first 10 identities** and visualize the features of training/validation data extracted from pretrained alexnet and your own model to 2D space with t-SNE, compare the results and explain the difference.
5. Implement your own method from scratch.

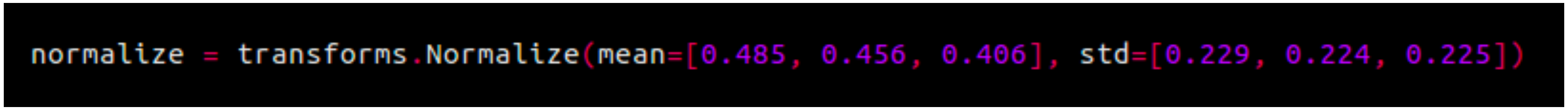
Dataset

- Description:
  - Images collected from VGG face dataset
  - Total 100 identities
  - Train/Validation images: 6987/1526
  - Image shape: 96 x 96 x 3 (height x width x channel)
- Pretrained model: AlexNet



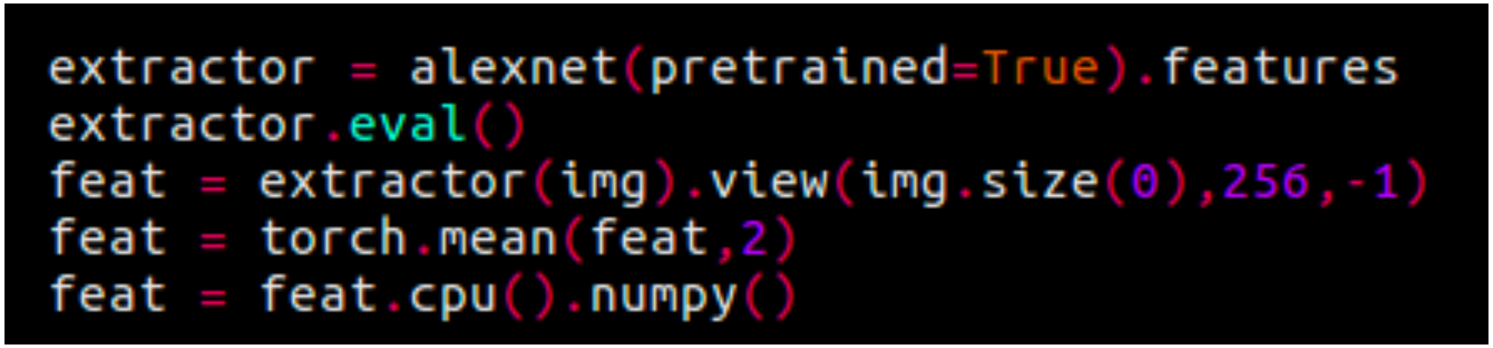
Extract feature with Alexnet

Dataloader: parameters of data normalization



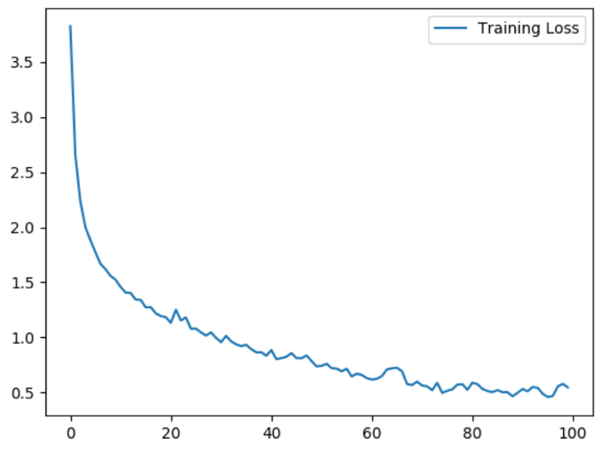
```
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

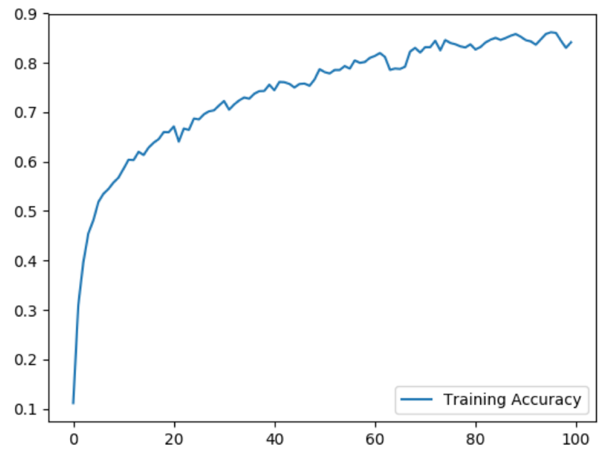
Extract feature with AlexNet and compute mean

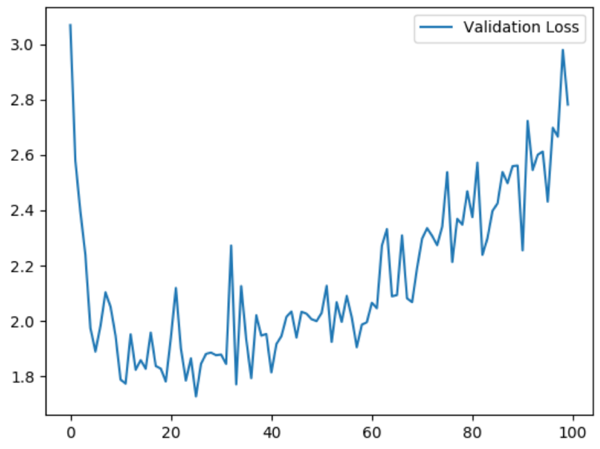


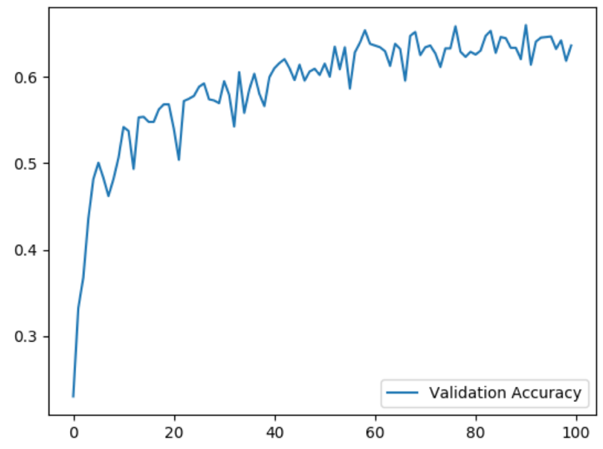
```
extractor = alexnet(pretrained=True).features
extractor.eval()
feat = extractor(img).view(img.size(0),256,-1)
feat = torch.mean(feat,2)
feat = feat.cpu().numpy()
```

Example

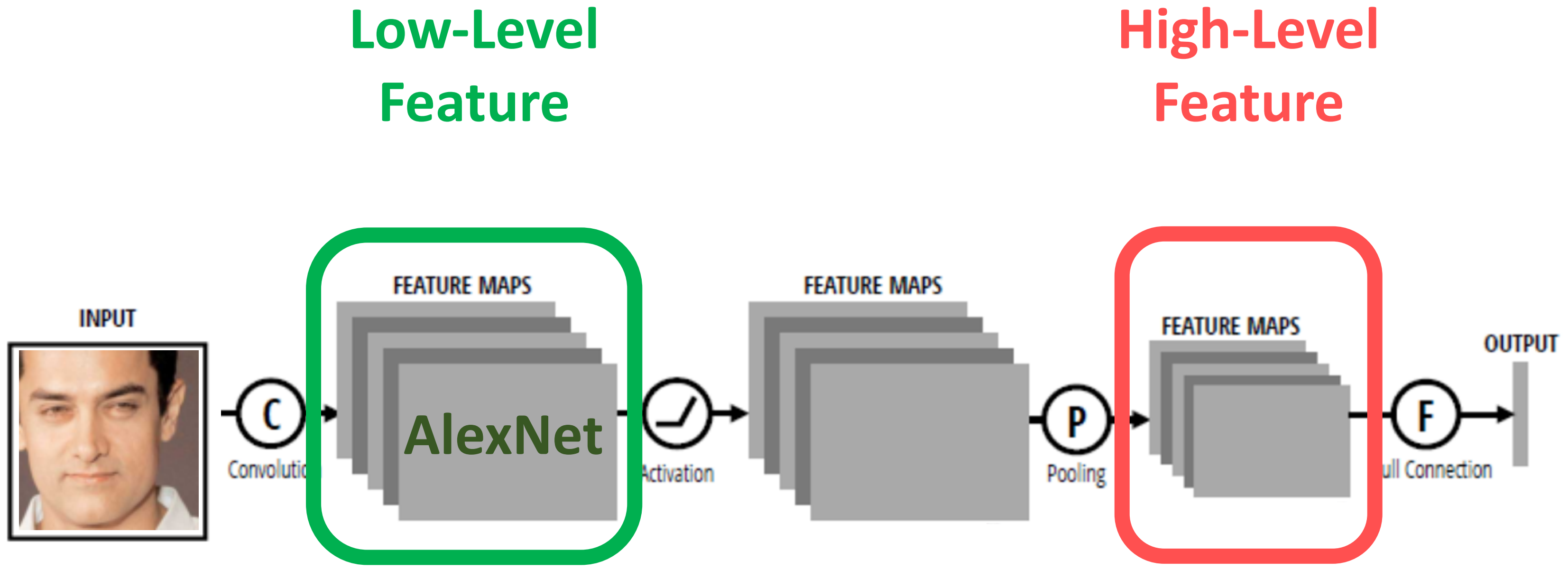

Training Loss
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
20
40
60
80
100


Training Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100

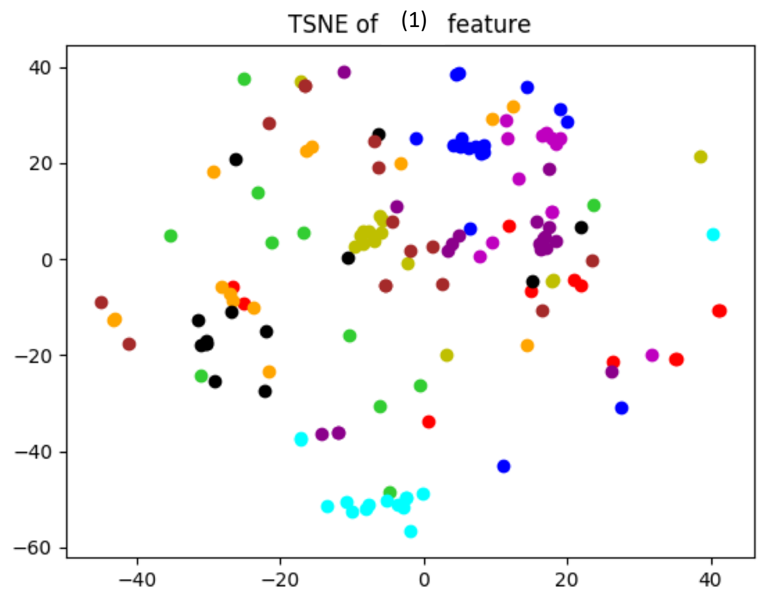

Validation Loss
3.0
2.8
2.6
2.4
2.2
2.0
1.8
0
20
40
60
80
100

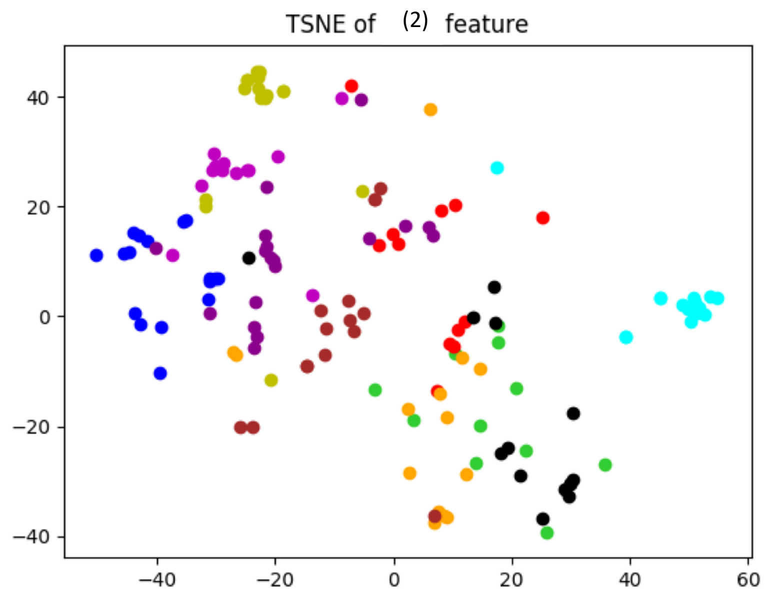

Validation Accuracy
0.6
0.5
0.4
0.3
0
20
40
60
80
100

Convolutional Neural Network



Examples of Feature Visualization





Regulations

Submission

- Your submission should include the following files:

  A **folder** named StudentID which contains
  - report_StudentID.pdf (e.g., report_R07654321.pdf)
  - pca.py
  - kmeans.py
  - train_mnist.py
  - test_mnist.py
  - baseline.py
  - Other python files and model files for hw2-4 (could be loaded by your python files)
- Do not upload the dataset
- Compress the **folder** in a zip file named StudentID.zip
  - e.g., R07654321.zip
- Submit to CEIBA
- Deadline: 11/12 11:00 p.m.

Submission (cont'd)

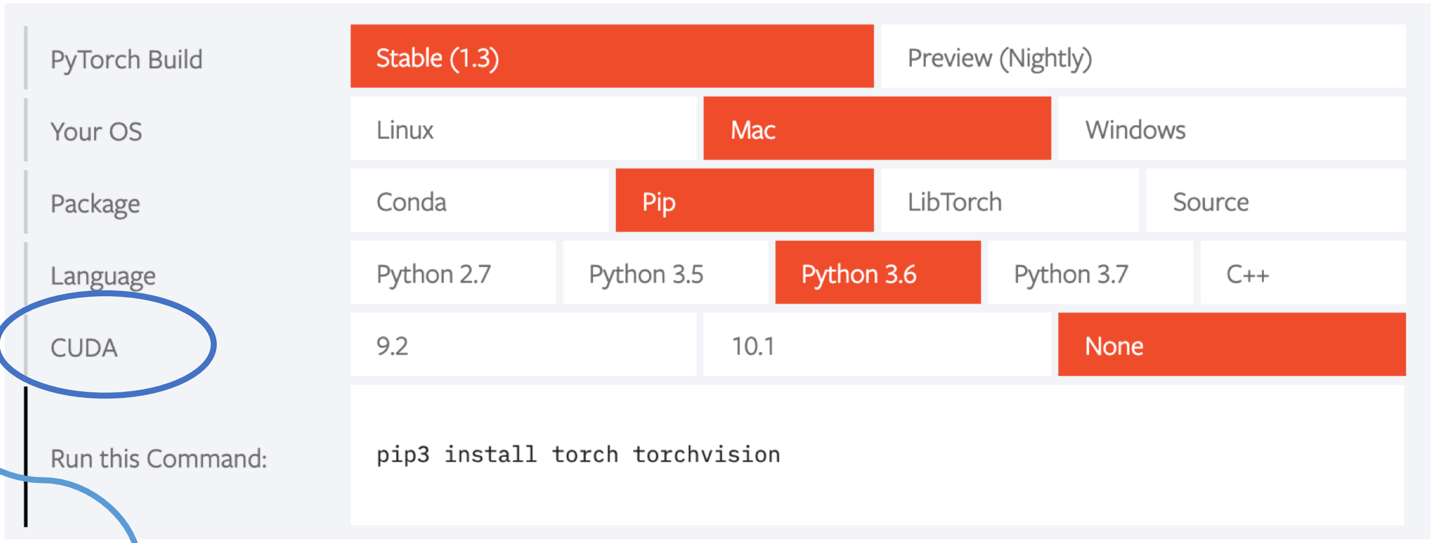
- If your model file cannot be uploaded to CEIBA due to the size limit (15 MB), you may upload it to other cloud service platforms (e.g., Dropbox). However, your python program should be able to download the model automatically.

Packages

- Python 3.5+
- Allowed packages
  - PyTorch 1.0.0+
  - pandas
  - numpy, scipy
  - opencv-python
  - matplotlib
  - scikit-learn (sklearn)
  - scikit-image (skimage), PIL, imageio, Python standard library
- Please ask TAs (via e-mail or on FB group) if you want to use any other packages

Installation

- Pytorch official website: https://pytorch.org/
- Choose the options base on your computer
- Mac: pip3 install torch torchvision

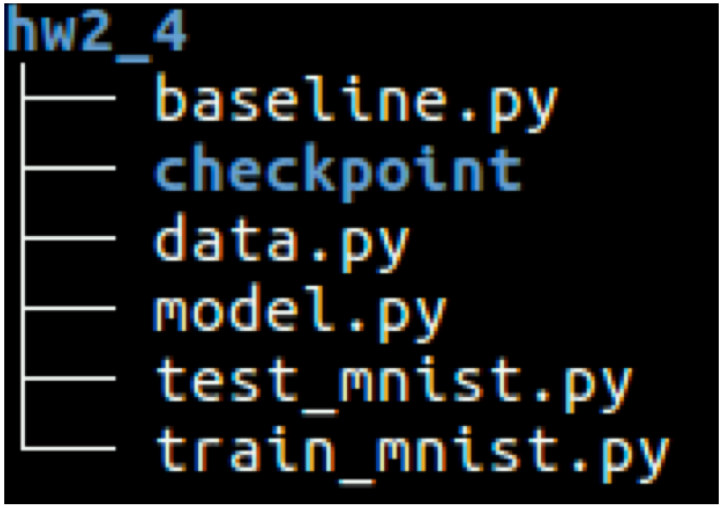


PyTorch Build	[✓ Stable (1.3)]	Preview (Nightly)			
Your OS	Linux	[✓ Mac]	Windows		
Package	Conda	[✓ Pip]	LibTorch	Source	
Language	Python 2.7	Python 3.5	[✓ Python 3.6]	Python 3.7	C++
CUDA	9.2	10.1	[✓ None]		
Run this Command:	pip3 install torch torchvision				

Choos “None” if you don’t have GPU

Data and Templates

- Data: link
- Report template: link
- Code:



```
hw2_4
├── baseline.py
├── checkpoint
├── data.py
├── model.py
├── test_mnist.py
└── train_mnist.py
```

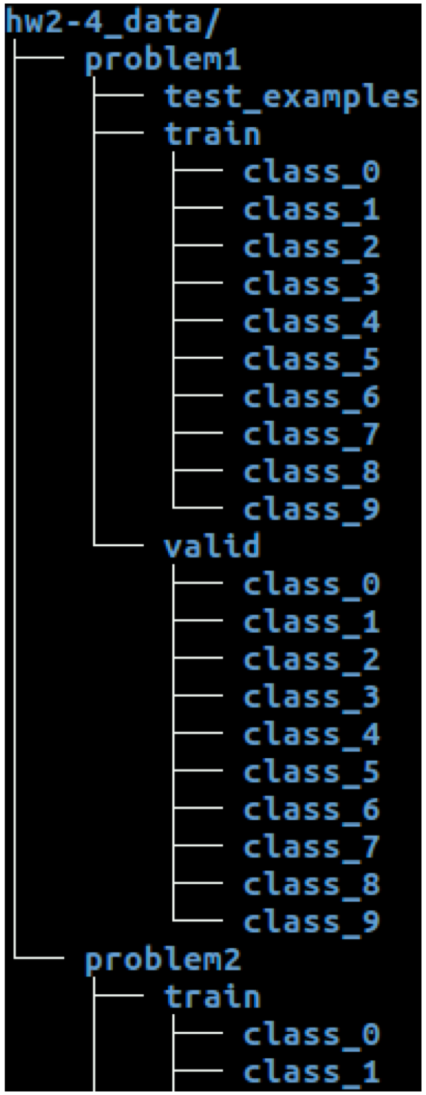
Execution for HW2-3

- TAs will run your code in the following manner:
  - python3 pca.py \$1 \$2
    - \$1: path of whole dataset
    - \$2: folder path of the output images including mean face and first five eigenfaces
    - E.g., python3 pca.py ./hw2-3_data ./pca_output/

  - python3 kmeans.py \$1 \$2
    - \$1: path of whole dataset
    - \$2: path of kmeans results visualization image
    - E.g., python3 kmeans.py ./hw2-3_data ./output_kmeans.png

Execution for HW2-4_1

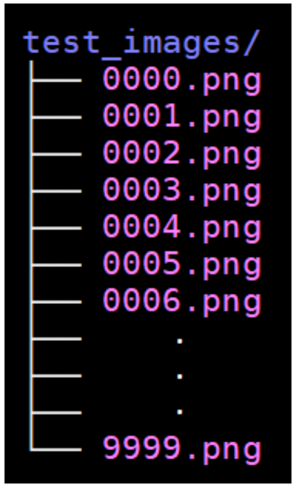
- TAs will run your code in the following manner:
- python3 train_mnist.py $1 $2
  - $1: directory of the hw2-4_1 data folder
  - $2: type of model (fully/conv)
  - E.g., python3 train.py ./hw2-4_data/problem1/
- python3 test_mnist.py $1 $2 $3
  - $1: directory of the testing images folder
  - $2: type of model (fully/conv)
  - $3: path folder of the output prediction file
  - E.g., python3 test.py ./hw2-4_data/problem1/test_images/ ./output/out.csv



```
hw2-4_data/
├── problem1
│   ├── test_examples
│   ├── train
│   │   ├── class_0
│   │   ├── class_1
│   │   ├── class_2
│   │   ├── class_3
│   │   ├── class_4
│   │   ├── class_5
│   │   ├── class_6
│   │   ├── class_7
│   │   ├── class_8
│   │   └── class_9
│   └── valid
│       ├── class_0
│       ├── class_1
│       ├── class_2
│       ├── class_3
│       ├── class_4
│       ├── class_5
│       ├── class_6
│       ├── class_7
│       ├── class_8
│       └── class_9
└── problem2
    ├── train
    │   ├── class_0
    │   ├── class_1
```

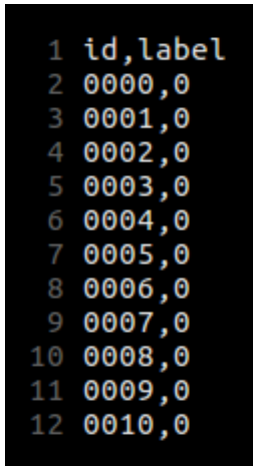
Execution for HW2-4_1

- Testing images folder include images named:
  - 0000.png , 0002.png , … , 9999.png



```
test_images/
├── 0000.png
├── 0001.png
├── 0002.png
├── 0003.png
├── 0004.png
├── 0005.png
├── 0006.png
├──    .
├──    .
├──    .
└── 9999.png
```

- Output prediction file format:
  - In csv format
  - First row:“id,label”
  - From second row: “<image_id>,<predicted_label>”



```
id,label
0000,0
0001,0
0002,0
0003,0
0004,0
0005,0
0006,0
0007,0
0008,0
0009,0
0010,0
```

Execution for HW2-4_2

- TAs will run your code in the following manner:
  - python3 baseline.py $1 $2
    - $1: directory of the hw2-4_2 data folder
    - $2: path of the output tsne figure(validation data)
    - E.g., python3 baseline.py ./hw2-4_data/problem2/ ./baseline_tsne.png