Assignment 2:

ML & DL Basics

Computer Vision

National Taiwan University

Fall 2018

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.
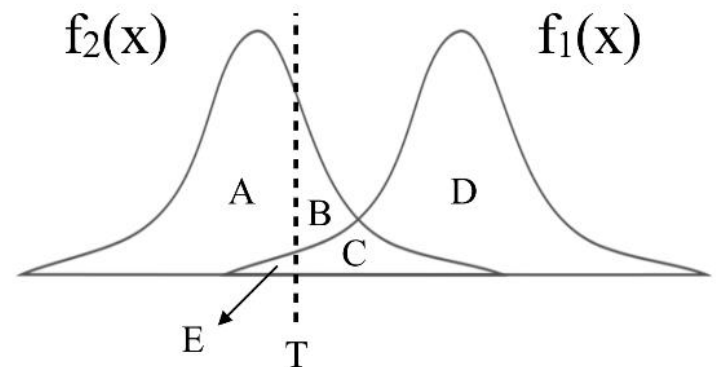
Problem 1(b)

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 answer that under what circumstances will the ROC curve fall on curve (b)?
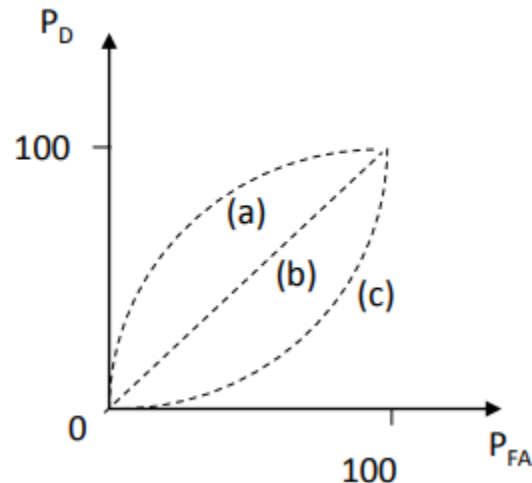
Part 2

PCA & LDA

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.

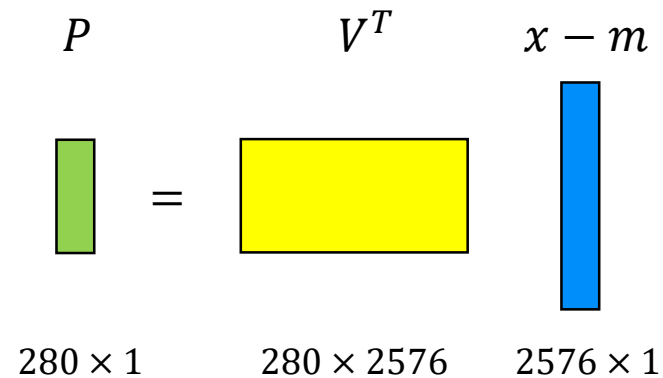Axis that describes the largest variation
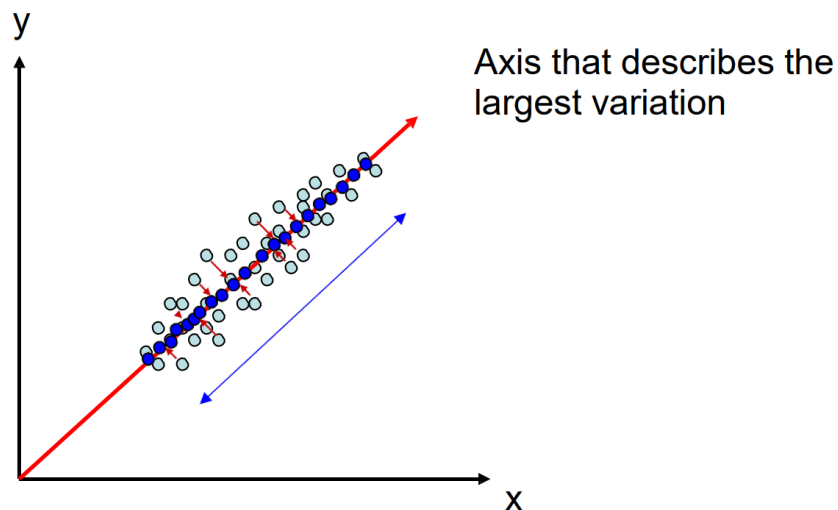
PCA – Eigenfaces

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

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

PCA – Image Reconstruction

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

- $P = V^T(x - m)$

  mean face

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

  $= Vp + m$

$$P \qquad\qquad V^T \qquad\qquad x - m$$

$$\text{[green]} \quad = \quad \text{[yellow]} \quad \text{[blue]}$$

$$280 \times 1 \qquad 280 \times 2576 \qquad 2576 \times 1$$

9

Linear Discriminant Analysis (LDA)

- Find projections to separate different classes.

Axis that describes the largest variation (or scatter)
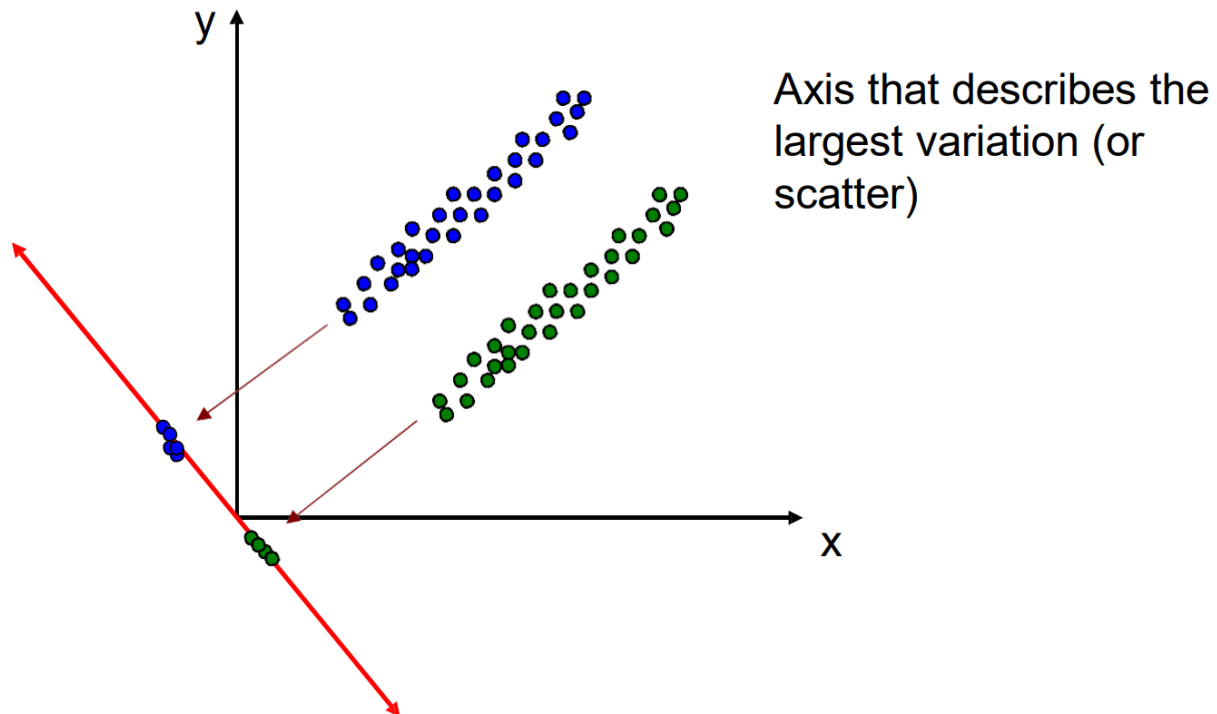
Fisherfaces

- Procedure
    1) Perform PCA and keep $(N - C)$ eigenvectors.
    2) Project data onto these $(N - C)$ eigenvectors.
       (S_w will be full rank)
    3) Perform LDA and compute the $(C - 1)$ projections in the lower-dimension $(N - C)$ subspace.

- Fisherfaces $=$ PCA $+$ LDA

$$S_w^{-1} S_B\, w = \lambda w$$

$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{N_i} (x_j - \mu_i)(x_j - \mu_i)^T$$

$$S_B = \sum_{i=1}^{C} (\mu_i - \mu)(\mu_i - \mu)^T$$

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 "$i\_j.png$" refers to "$person_i\_image_j$")

- 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


```
hw2-2_data/
├── 10_10.png
├── 10_1.png
├── 10_2.png
├── 10_3.png
├── 10_4.png
├── 10_5.png
├── 10_6.png
├── 10_7.png
├── 10_8.png
├── 10_9.png
├── 1_10.png
├── 11_10.png
├── 11_1.png
    ⋮
    ⋮
```

Problem 2(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 $person_8\_ image_6$, 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 $person_8\_ image_6$. 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.

t-SNE reference: http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

Problem 2(b) – LDA

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

1. Implement LDA and plot first 5 Fisherfaces.

2. Use t-SNE to visualize the distribution of the projected testing data, which has the dimension of 100.

Problem 2(c)

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.

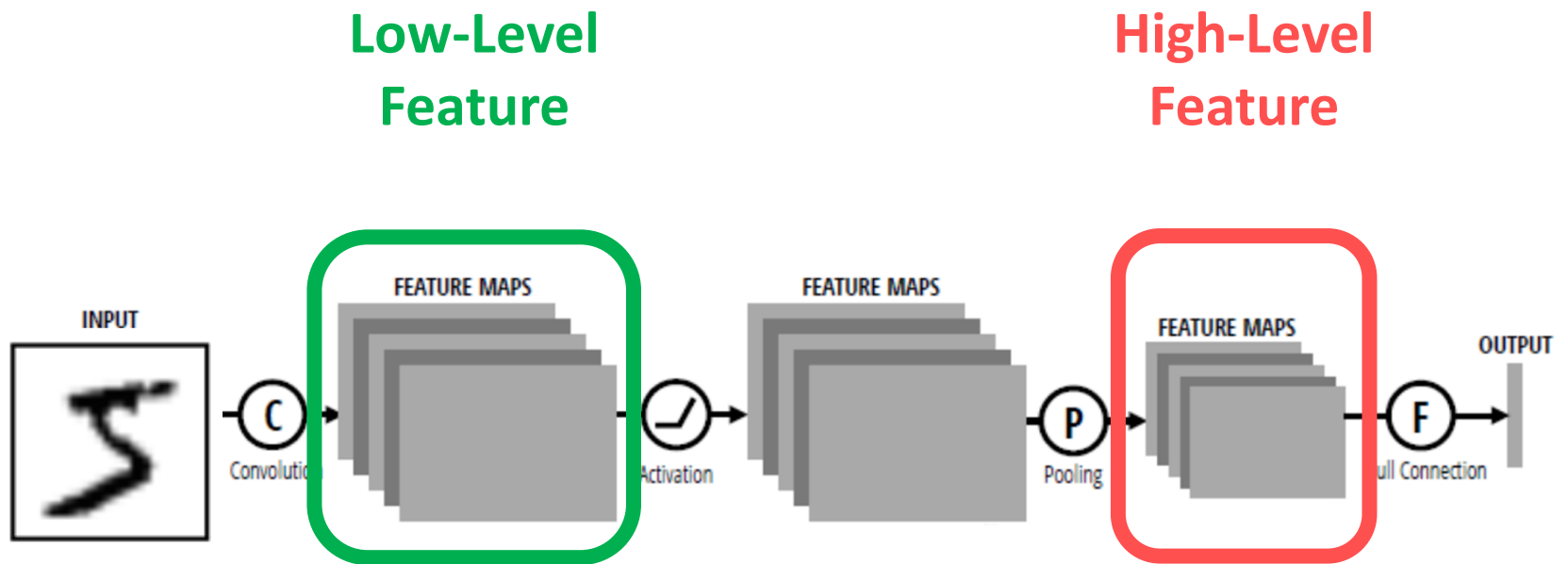For simplicity, the choices for such hyper-parameters are:

k = {1, 3, 5} 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. Please apply the above comparing method on both PCA and LDA, and explain your results.

Part 3

Object Recognition using CNN
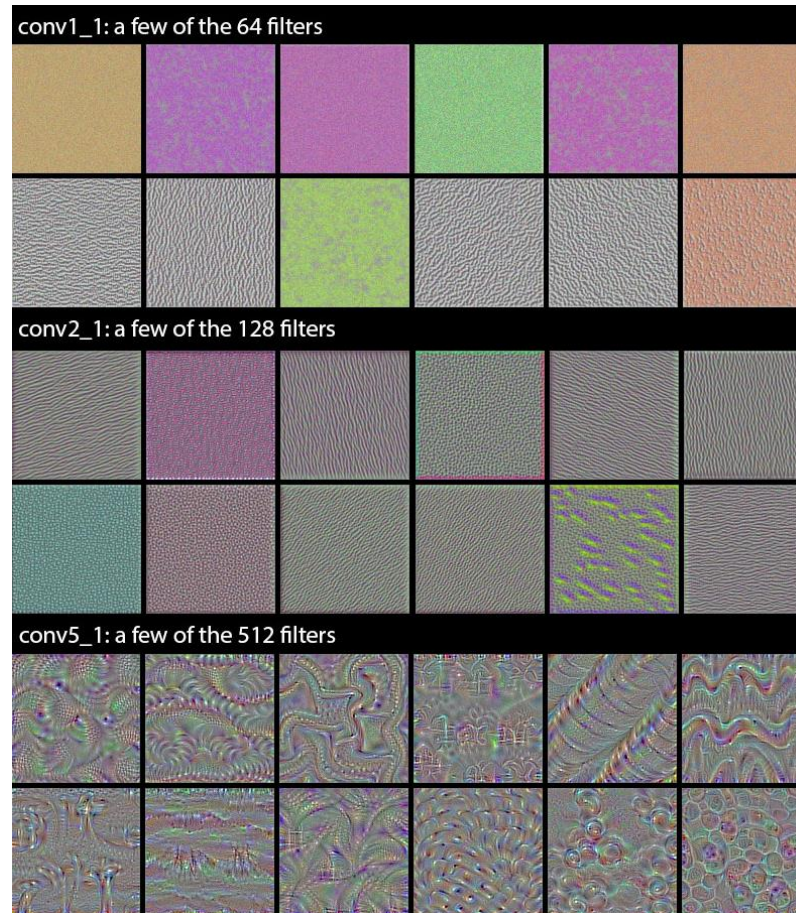
Convolutional Neural Network

Filter Visualization

conv1_1: a few of the 64 filters

conv2_1: a few of the 128 filters

conv5_1: a few of the 512 filters

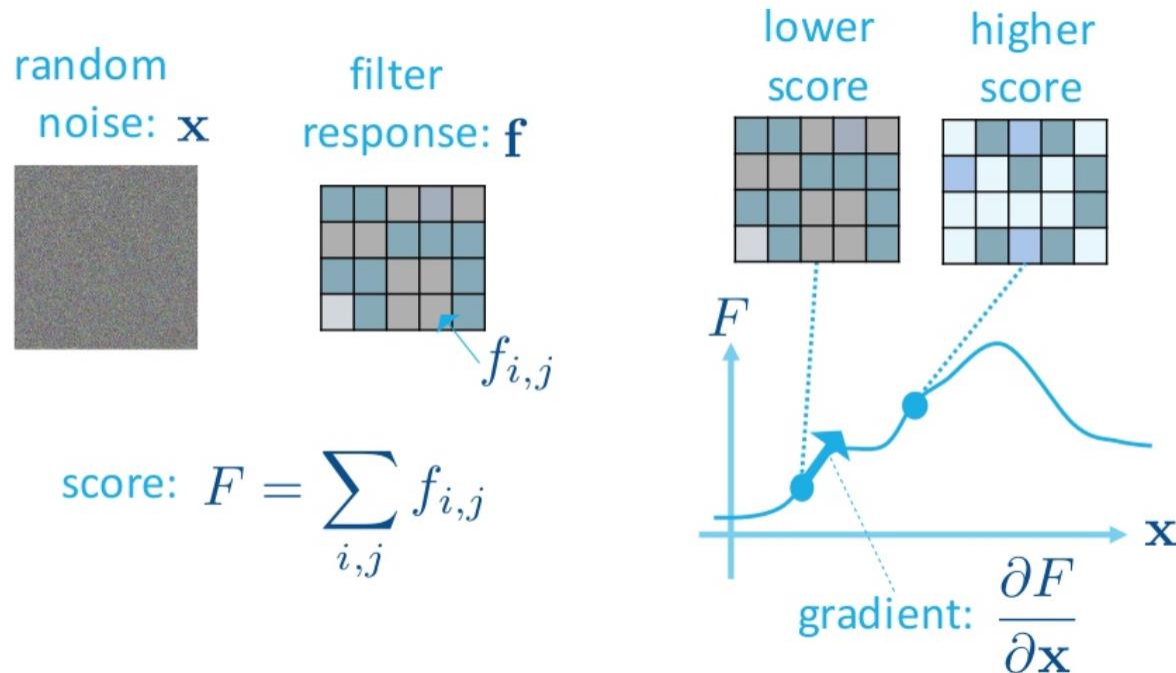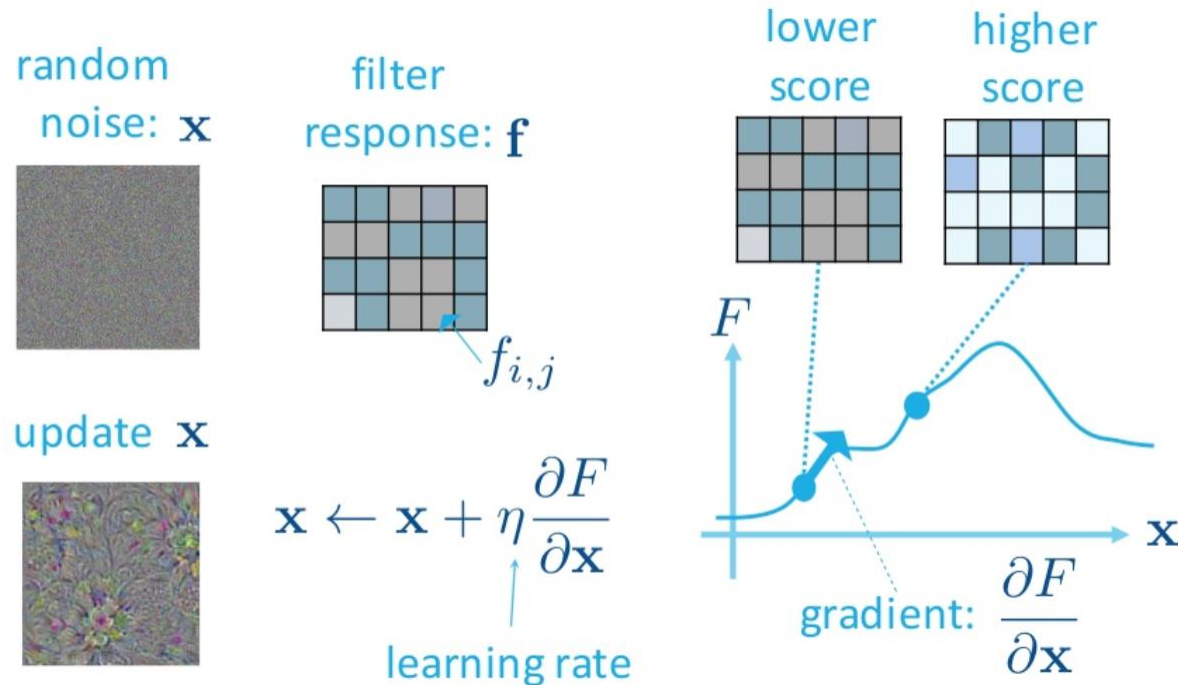https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

Filter Visualization

- Gradient ascent method
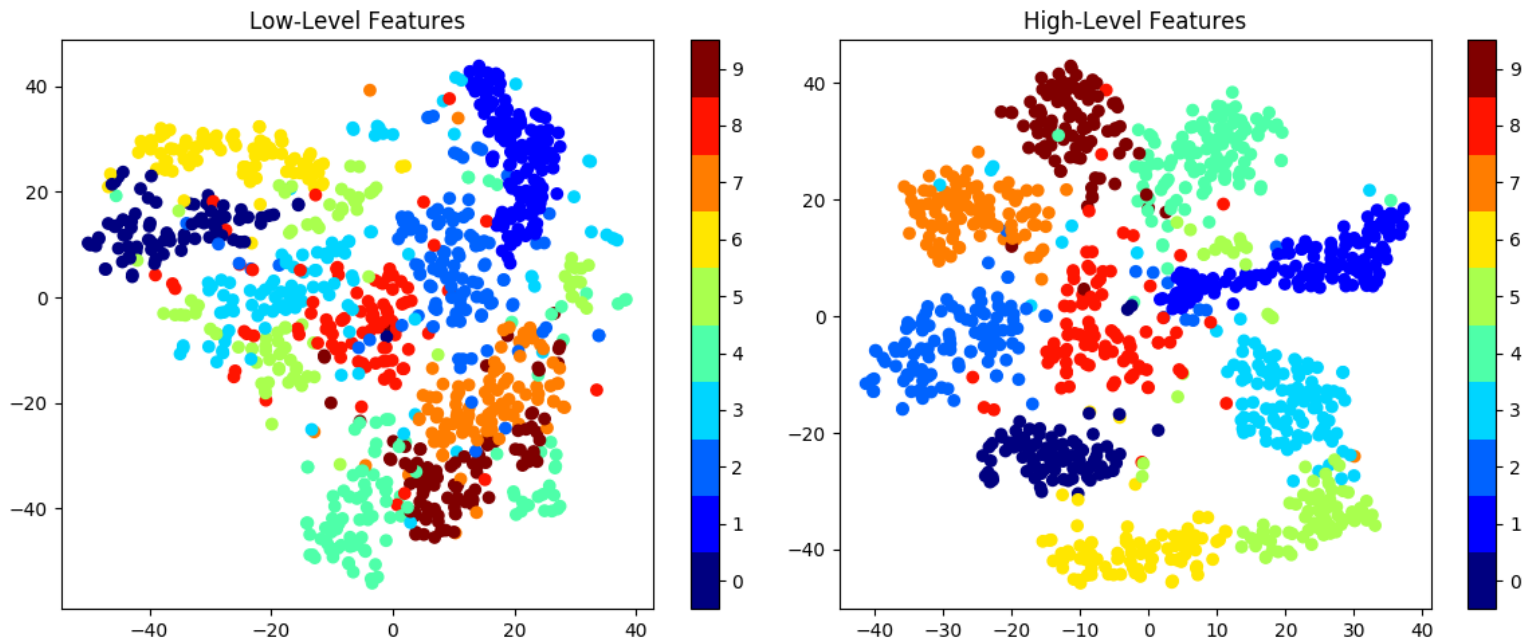 - Magnify the response of the specific filter in the specific layer

Filter Visualization

- Gradient ascent method
  - Magnify the response of the specific filter in the specific layer

Feature Visualization

- Visualize the features to 2D space with t-distributed stochastic neighbor embedding (t-SNE)
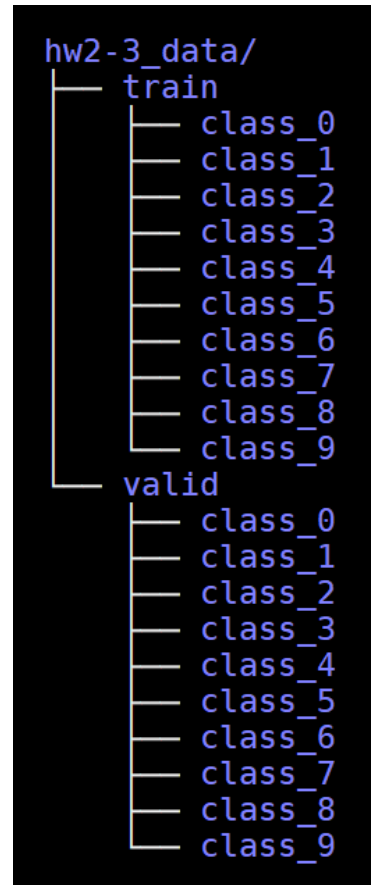
Dataset

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

  - Training images
    - 5000 images for each class, 50000 images in total
    - Images: 0000.png , … , 4999.png

  - Validation images
    - 1000 images for each class, 10000 images in total
    - Images: 5000.png , … , 5999.png


```
hw2-3_data/
├── 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
```

Problem 3

a. Build a CNN model and train it on the given dataset. Show the architecture of your model in the report.

b. Report your training / validation accuracy, and plot the learning curve (loss, accuracy) of the training process. The prediction accuracy should pass the following baselines.

- Validation accuracy: 98 % , Testing accuracy: 96 %
- You may consider using a LeNet-5 as the CNN model.

c. Visualize filters on the first and the last convolutional layers.

d. Visualize high-level and low-level features extracted from different layers, and explain what you have observed from the two t-SNE plots.

LeNet-5

softmax

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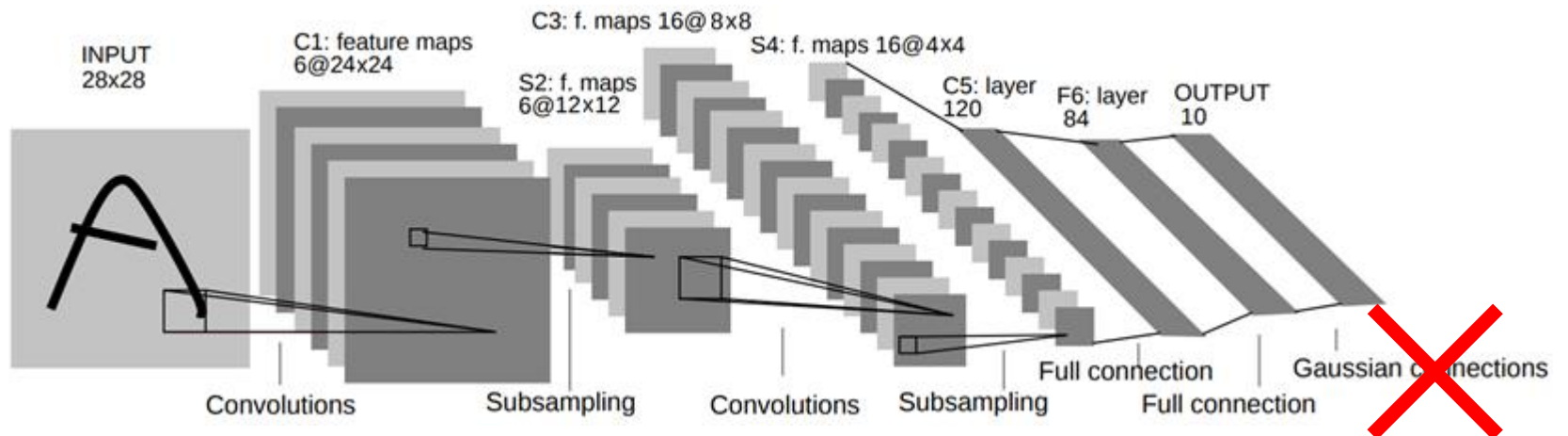
Regulations

Submission

- Your submission should include the following files.
  - report_StudentID.pdf (e.g., report_R07654321.pdf)
  - hw2-2_pca.py
  - hw2-2_lda.py
  - hw2-3_train.py
  - hw2-3_test.py
  - model files for hw2-3 (could be loaded by your python files)
- Do not upload the dataset.
- Compress all above files in a zip file named StudentID.zip
  - e.g., R07654321.zip
- Submit your zip file to CEIBA.
- Deadline: 11/13 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.

Data & Report Template

- Data download link: hw2_data.zip

- Report template: link

```
hw2_data.zip
├── hw2-2_data
└── hw2-3_data
    ├── 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
```
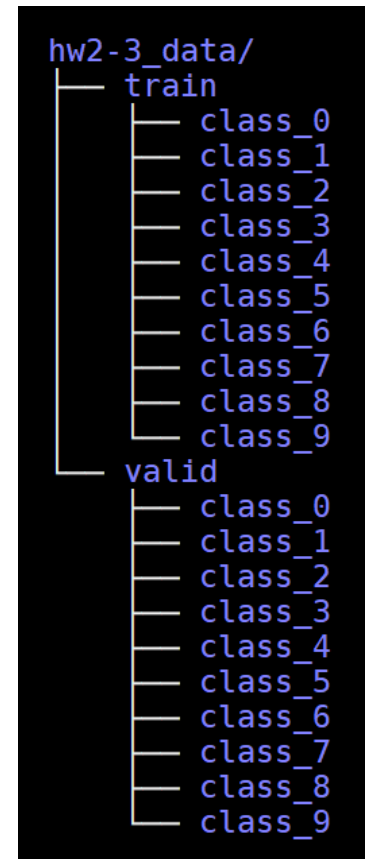
Packages

- Python 3.5+
- Allowed packages
  - Keras 2.2.2
  - TensorFlow 1.10.1
  - PyTorch 0.4.1
  - pandas
  - numpy, scipy
  - opencv-python
  - 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.

Execution for HW2-2

- TAs will run your code in the following manner.
    - python3 hw2-2_pca.py $1 $2 $3
        - $1: path of whole dataset
        - $2: path of the input testing image
        - $3: path of the output testing image reconstruct by all eigenfaces
        - E.g., python3 hw2-2_pca.py ./hw2/hw2-2_dataset
          ./hw2/test_image.png ./output_pca.png

    - python3 hw2-2_lda.py $1 $2
        - $1: path of whole dataset
        - $2: path of the first 1 Fisherface
        - E.g., python3 hw2-3_lda.py ./hw2/hw2-2_dataset ./output_fisher.png

Execution for HW2-3

- TAs will run your code in the following manner.
  - python3 hw2-3_train.py $1
    - $1: directory of the hw2-3_data folder
    - E.g., python3 hw2-3_train.py ./hw2/hw2-3_data/

```
hw2-3_data/
├── 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
```

Execution for HW2-3 (cont'd)

- TAs will run your code in the following manner.
  - python3 hw2-3_test.py $1 $2
    - $1: directory of the testing images folder
    - $2: path of the output prediction file
    - E.g., python3 hw2-3_test.py ./test_images/ ./output.csv

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

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

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

```
1  id,label
2  0000,0
3  0001,0
4  0002,0
5  0003,0
6  0004,0
7  0005,0
8  0006,0
9  0007,0
10 0008,0
11 0009,0
12 0010,0
```