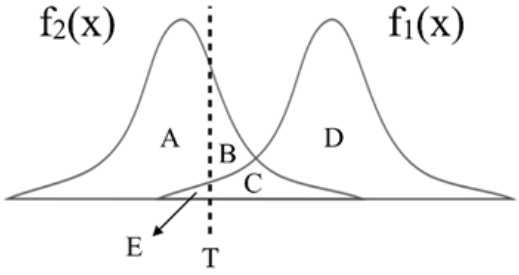
Up Computer Vision: from Recognition to Geometry
HW2

Name: 陳竣宇 Department: 資工四 Student ID: B05902058

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. (6%)



Ans:

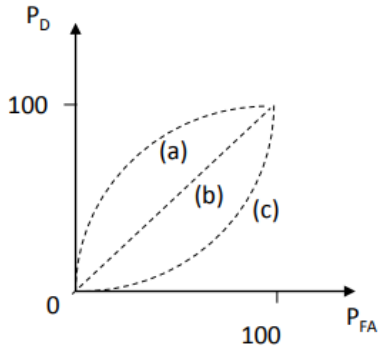
(1) False Positive: B+C

根據定義，False Positive 為真實值是 negative 但被 classify 為 positive 的機率，因此為 f2(x)在 T 之後的區域

(2) False Negative: E

根據定義，False Negative 為真實值是 positive 但被 classify 為 negative 的機率，因此為 f1(x)在 T 之前的區域

(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)? (6%)



Ans:

(1) 曲線(a)是合理的 ROC curve 而曲線(c)則否。以上題之 PDF 來判斷，假設 T 為負無窮大，則此時的p_D和p_{FA}皆為 100%，也就是 ROC curve 上(100, 100)之點，接著嘗試將 T 往正方向移動，可以推知p_{FA}必會下降的較p_D快，因為p_{FA}在原圖中較負的位置。

(2) 曲線(b)代表無論 T 是多少p_D和p_{FA}都是相同的值，當 positive 和 negative 具有相同的 PDF 時會呈現此種情況，也就是給定任意 X 分類為 positive 和 negative 的機率相同(50%)

Problem 2

(a) Given a convolutional layer which contains:

n kernels with size $k * k * n_{in}$
padding size p,
stride size (s,s).

With input feature size $W * W * n_{in}$, calculate the size of output feature $W_{out} * W_{out} * n_{out}$.

(i) W_out=? (2%)
(ii) n_out =? (2%)

Ans:

(i) $\left\lfloor \frac{W+2*p-k}{s} \right\rfloor + 1$
(ii) n

(b) k=5, s=2, p=1, n=256, W=64, calculate the number of parameters in the convolutional layer (4%)

Ans:

$3 * (5 * 5) * 256 = 19200$

Problem 3 (report 35% + code 5%)

(a) PCA (15%)
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. (5%)

Ans:

- Mean face:

- First 5 eigenfaces:

2. Take $\boldsymbol{person}_8\_\ \boldsymbol{image}_6$, 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 $\boldsymbol{person}_8\_\ \boldsymbol{image}_6$. Plot these reconstructed images with the corresponding MSE values in the report. (5%)

Ans:

- n = 5:

 , MSE = 89.80667701863354

- n = 50:

 , MSE = 56.19332298136646

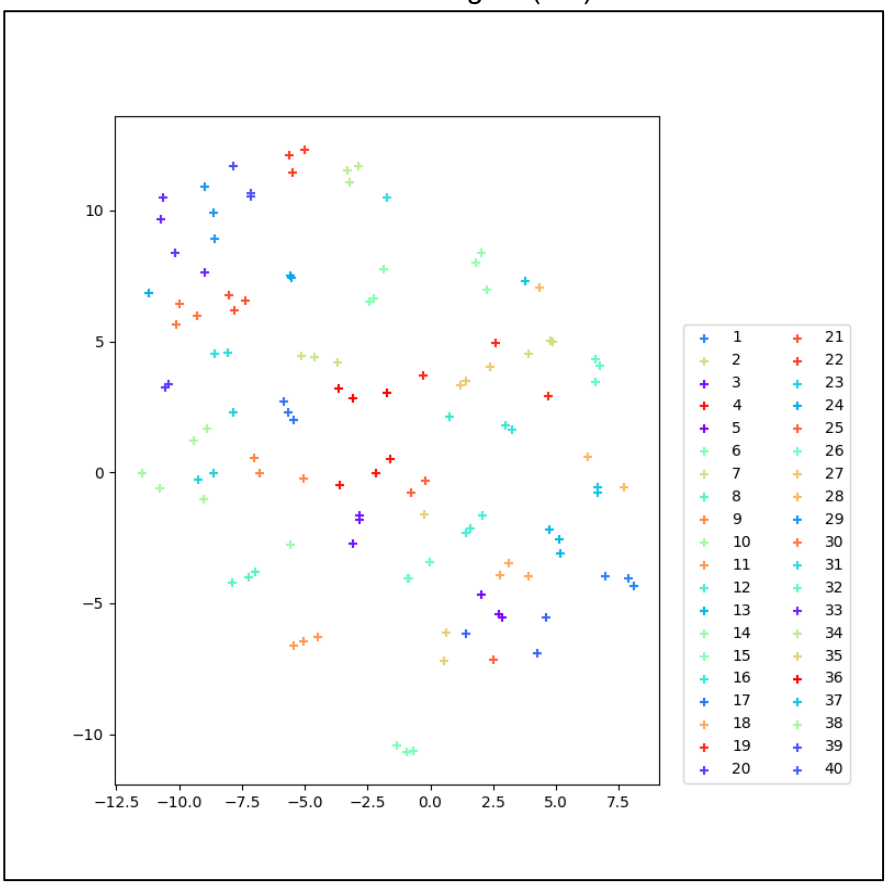
- n = 150:

 , MSE = 32.576863354037265

- n = all:

 , MSE = 0.5139751552795031

3. Reduce the dimension of the image in testing set to dim = 100. Use t-SNE to visualize the distribution of test images. (5%)



4. (bonus 5%) Implement the Gram Matrix trick for PCA. Compare the two reconstruction images from standard PCA/Gram Matrix process. If the results are different, please explain the reason. Paste the main code fragment(screenshot) on the report with discussion.

Ans:

- 在 n = 5, 50, 150 時使用 gram matrix 重建之 image 的 MSE 和使用 standard PCA 是相同的，而 n = all 時則會因為浮點數誤差產生一點微小的差距(0.47 v.s. 0.51)

```
def getPCA(self, use_gramMatrix):
    # Eigen decomposition
    if not use_gramMatrix:
        u, s, v = np.linalg.svd(self.data.T, full_matrices=False)
        return u
    else:
        tmp = self.data.dot(self.data.T)
        u, s, v = np.linalg.svd(tmp.T, full_matrices=False)
        u = self.data.T.dot(u)
        for i in range(u.shape[1]):
            u[:, i] = u[:, i] / np.linalg.norm(u[:, i])
        return u
```

(b) k-NN (10%)

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, 10, 39}.

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.

Ans:

- Cross-validation results(average score):
  - n = 3:
    - k = 1: 0.6833333333333332
    - k = 3: 0.5930555555555556
    - k = 5: 0.5152777777777778
  - n = 10:
    - k = 1: 0.8513888888888889
    - k = 3: 0.7458333333333332
    - k = 5: 0.6777777777777777
  - n = 39:
    - k = 1: 0.9236111111111112
    - k = 3: 0.8486111111111111
    - k = 5: 0.7763888888888889
- (best k, best n) = (1, 39)。意思為用 PCA 把 data 降到 39 維，KNN 以和周圍最近的 1 個點來 classify 可以得到最好的 cross validation score

- Recognition rate on testing test = 0.958

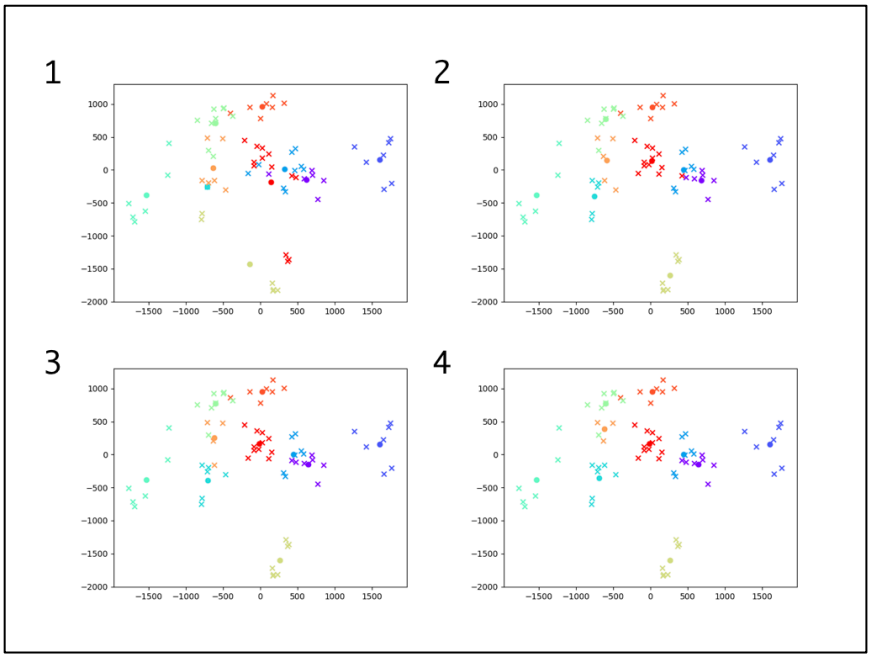
(c) K-means (10%)

Reduce the dimension of the images in the first 10 class of training set to dim = 10 using PCA. Implement the k-means clustering method to classify the these images.

1. Please use weighted Euclidean distance to implement the k-means clustering. The weight of the best 10 eigenfaces is

   [0.6, 0.4, 0.2, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]

2. Please visualize the features and the centroids to 2D space with the first two eigenfaces for each iteration in k-means clustering (up to 5 images). (5%)



3. Compare the results of K-means and ground truth. (5%)

Ans:

比對 K-means 的結果和 ground truth 的方式是在分群後以每個群 label 的頻率最大值為此群的 class，然後統計每個 data 的 label 是否為此群的 class，最終得出 accuracy 大約在 0.8 之間震盪。

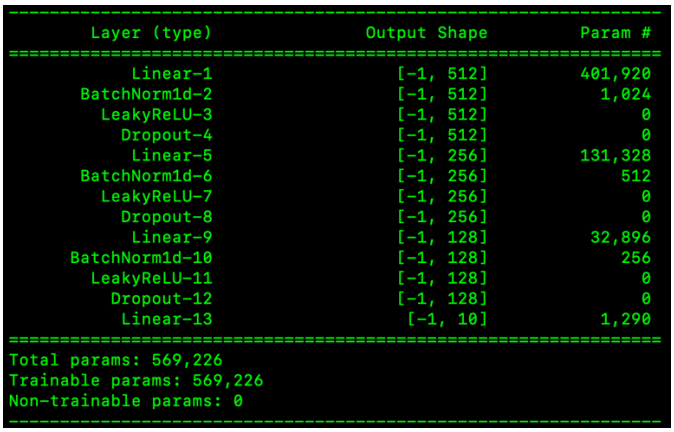
Problem 4 (report 30% + baseline 10%)

(a) MNIST Classification (20%)

1. Build a CNN model and a Fully-Connected Network and train them on the MNIST dataset. Show the architecture of your models and the correspond parameters amount in the report. (5%)

Ans:

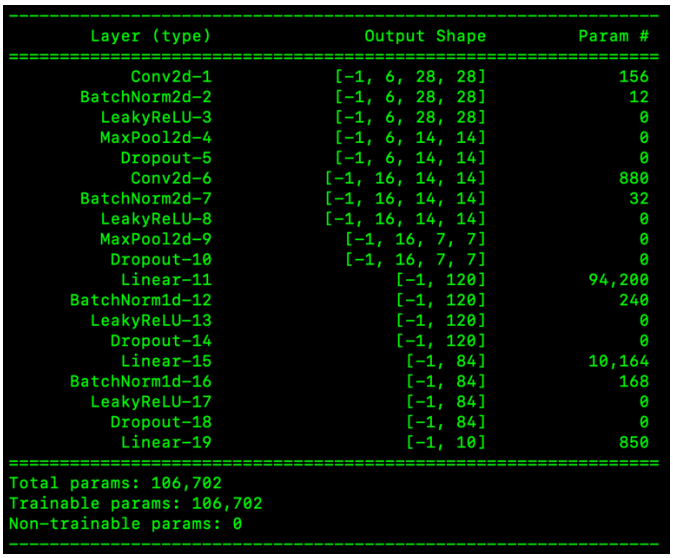
- FC:



Layer (type)	Output Shape	Param #
Linear-1	[-1, 512]	401,920
BatchNorm1d-2	[-1, 512]	1,024
LeakyReLU-3	[-1, 512]	0
Dropout-4	[-1, 512]	0
Linear-5	[-1, 256]	131,328
BatchNorm1d-6	[-1, 256]	512
LeakyReLU-7	[-1, 256]	0
Dropout-8	[-1, 256]	0
Linear-9	[-1, 128]	32,896
BatchNorm1d-10	[-1, 128]	256
LeakyReLU-11	[-1, 128]	0
Dropout-12	[-1, 128]	0
Linear-13	[-1, 10]	1,290

Total params: 569,226
Trainable params: 569,226
Non-trainable params: 0

- CNN:



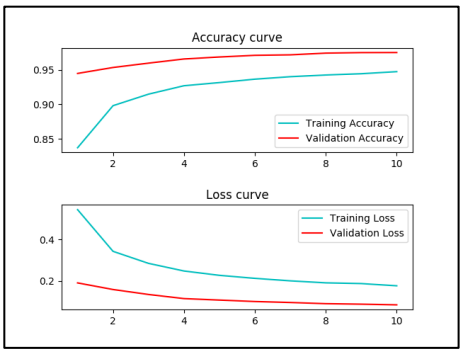
Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 6, 28, 28]	156
BatchNorm2d-2	[-1, 6, 28, 28]	12
LeakyReLU-3	[-1, 6, 28, 28]	0
MaxPool2d-4	[-1, 6, 14, 14]	0
Dropout-5	[-1, 6, 14, 14]	0
Conv2d-6	[-1, 16, 14, 14]	880
BatchNorm2d-7	[-1, 16, 14, 14]	32
LeakyReLU-8	[-1, 16, 14, 14]	0
MaxPool2d-9	[-1, 16, 7, 7]	0
Dropout-10	[-1, 16, 7, 7]	0
Linear-11	[-1, 120]	94,200
BatchNorm1d-12	[-1, 120]	240
LeakyReLU-13	[-1, 120]	0
Dropout-14	[-1, 120]	0
Linear-15	[-1, 84]	10,164
BatchNorm1d-16	[-1, 84]	168
LeakyReLU-17	[-1, 84]	0
Dropout-18	[-1, 84]	0
Linear-19	[-1, 10]	850

Total params: 106,702
Trainable params: 106,702
Non-trainable params: 0

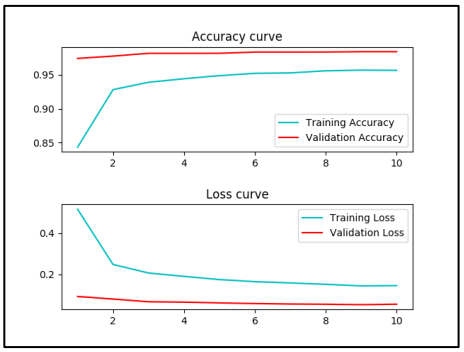
2. Report your training / validation accuracy, and plot the learning curve (loss, accuracy) of the training process. (figure 5%+ baseline 5%)

Ans:

- FC: 0.947 / 0.975



- CNN: 0.957 / 0.984



3. Compare the results of both models and explain the difference. (5%)

Ans:

由前幾題的結果可以看出 CNN 在 MNIST 數字辨識這個 task 上不僅 performance 較優，而參數量也比 fully-connected network 來的少。

課堂上提到的 Local Connectivity 和 Weight Sharing 等概念很好地體現在這個 task 上。除了參數量的明顯降低外，我認為表現較佳原因是因為 CNN 能有效的取得圖片的局部性質，也就是相鄰 pixel 之間的關係資訊。

(b) Face Recognition (20%)

1. Extract image feature using pytorch pretrained alexnet and train a KNN classifier to perform human face recognition on the given dataset. Report the validation accuracy. (5%)

Ans:

- validation accuracy = 0.520315

2. Build your own model and train it on the given dataset to surpass the accuracy of previous stage. Show and explain the architecture of your model and report the validation accuracy. Paste the main code fragment(screenshot). (5%)

Ans:

- validation accuracy = 0.874

- 我的 CNN model 包含了 4 層的 convolution layer，每層 layer 之間皆包括了 LeakyReLU / BatchNormalize / Dropout layer。kernel size 比照 AlexNet，為(5, 5), (3, 3), (3, 3), (3, 3)，Dropout 則是隨著層數逐漸提升。跑完 convolution layer 之後就把圖壓成一維後接兩層 fully-connected layer，最後就是再接一層 softmax 來產生 100 個 class 的分類結果。

  - Model architecture

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 64, 96, 96]           4,864
       BatchNorm2d-2          [-1, 64, 96, 96]             128
         LeakyReLU-3          [-1, 64, 96, 96]               0
         MaxPool2d-4          [-1, 64, 48, 48]               0
           Dropout-5          [-1, 64, 48, 48]               0
            Conv2d-6         [-1, 128, 48, 48]          73,856
       BatchNorm2d-7         [-1, 128, 48, 48]             256
         LeakyReLU-8         [-1, 128, 48, 48]               0
         MaxPool2d-9         [-1, 128, 24, 24]               0
          Dropout-10         [-1, 128, 24, 24]               0
           Conv2d-11         [-1, 512, 24, 24]         590,336
      BatchNorm2d-12         [-1, 512, 24, 24]           1,024
        LeakyReLU-13         [-1, 512, 24, 24]               0
        MaxPool2d-14         [-1, 512, 12, 12]               0
          Dropout-15         [-1, 512, 12, 12]               0
           Conv2d-16         [-1, 512, 12, 12]       2,359,808
      BatchNorm2d-17         [-1, 512, 12, 12]           1,024
        LeakyReLU-18         [-1, 512, 12, 12]               0
        MaxPool2d-19           [-1, 512, 6, 6]               0
          Dropout-20           [-1, 512, 6, 6]               0
           Linear-21                 [-1, 1024]      18,875,392
      BatchNorm1d-22                 [-1, 1024]           2,048
        LeakyReLU-23                 [-1, 1024]               0
          Dropout-24                 [-1, 1024]               0
           Linear-25                  [-1, 512]         524,800
      BatchNorm1d-26                  [-1, 512]           1,024
        LeakyReLU-27                  [-1, 512]               0
          Dropout-28                  [-1, 512]               0
           Linear-29                  [-1, 100]          51,300
================================================================
Total params: 22,485,860
Trainable params: 22,485,860
Non-trainable params: 0
----------------------------------------------------------------
```

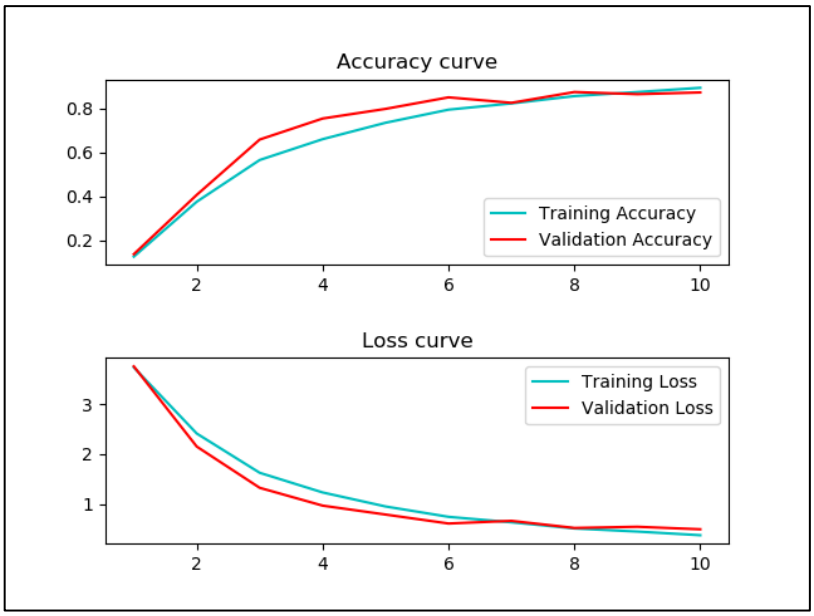
  - Main code fragment

```
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.9)

for epoch in range(num_epoch):
    print('[Epoch: {:d}]'.format(epoch))

    # Record the information of correct prediction and loss
    correct_cnt, total_loss, total_cnt = 0, 0, 0

    ## Training
    model.train()
    print('[Training]...')
    for batch, (img, label) in enumerate(train_loader, 1):
        # Set the gradients to zero (left by previous iteration)
        optimizer.zero_grad()
        # Put input tensor to GPU if it's available
        img, label = img.to(device), label.to(device)
        # Forward input tensor through your model
        out = model(img)
        # Calculate loss
        loss = criterion(out, label)
        # Compute gradient of each model parameters base on calculated loss
        loss.backward()
        # Update model parameters using optimizer and gradients
        optimizer.step()

        # Calculate the training loss and accuracy of each iteration
        total_loss += loss.item()
        _, pred_label = torch.max(out, 1)
        total_cnt += img.size(0)
        correct_cnt += (pred_label == label).sum().item()

        # Show the training information
        if batch % 5 == 0 or batch == len(train_loader):
            acc = correct_cnt / total_cnt
            ave_loss = total_loss / batch
            print ('Training batch index: {}, train loss: {:.6f}, acc: {:.3f}'.format(
                batch, ave_loss, acc))

    # Store acc && loss to plot
    Acc_train.append(acc)
    Loss_train.append(ave_loss)
```

3. Plot the learning curve of your training process(training/validation loss/accuracy) in stage 2, explain the result you observed. (5%)

Ans:

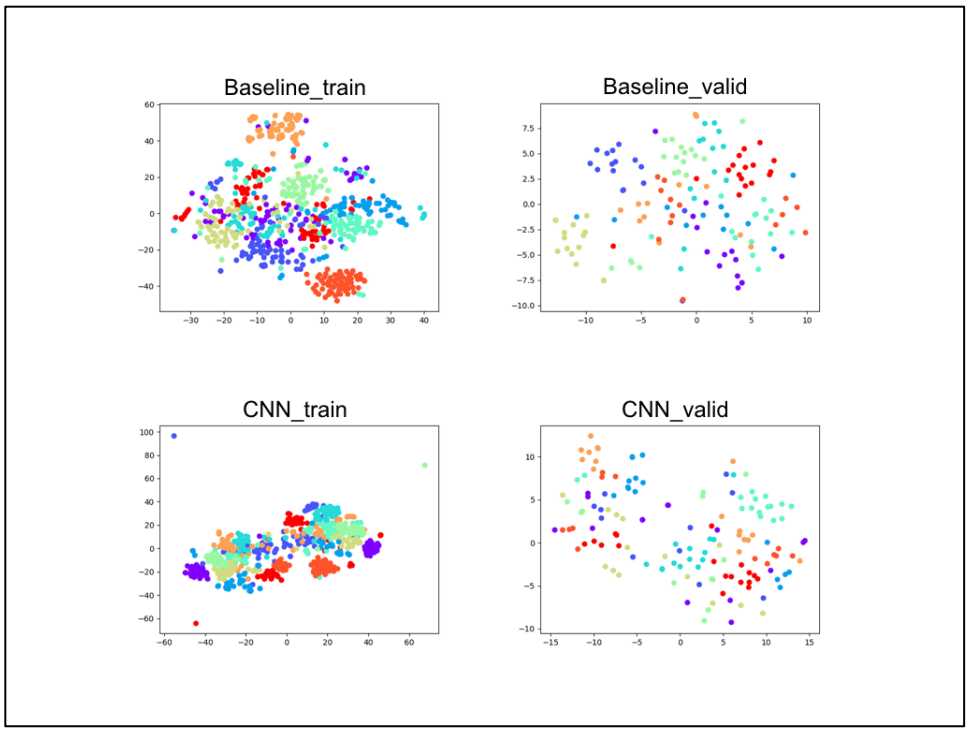
- 在 training process 一開始 loss 因為 gradient descent 降的很快，中期之後就慢慢減緩，因為已經接近 local minima。
- 沒有明顯的 overfitting 現象，原因可能是因為沒有跑太多 epoch，也可能是在每層 layer 之間都加了 Dropout 的關係。



4. Pick the first 10 identities from dataset. Visualize the features of training/validation data you extract from pretrained alexnet and your own model to 2D space with t-distributed stochastic neighbor embedding (t-SNE), compare the results and explain the difference. (5%)

Ans:

- 兩種方式都可以 extract 出有用的 information 來區分 data。
- 在使用 AlexNet feature extractor 和我的 CNN model 來抽取特徵的 task 上我覺得兩者相差不多甚至 AlexNet feature extractor 的表現還比較好，但是因為我的 CNN model 在跑完 convolution layer 後 MLP layer 直接使用前面的 feature 來 train，因此 performance 就比接 KNN 來的好很多。


Baseline_train
Baseline_valid
CNN_train
CNN_valid