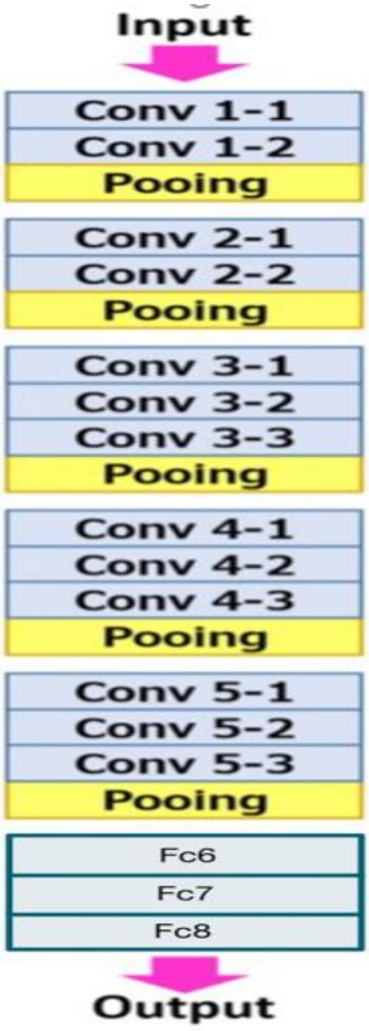
DLCV

Hw1

B10901187 電機三 鍾閎凱

Problem 1: Image Classification

1. Model A(VGG13) architecture



Using the VGG13 for model A, five blocks all contain(2 * (conv + ReLU) + max_pooling), and in the end add three fc layer to compute the output

2. Acc for Model A & B

	Model A(VGG)	Model B(Resnet152)
Acc	0.6515	0.8981

3. Implementation Details of Model A

Training settings:

1. Data augmentation: [RandomCrop, RandomRotation, RandomHorizontalFlip, Normalize]
2. Number of epoch = 500
3. Batch size = 128
4. lr = 0.001
5. loss_function = CrossEntropy
6. optimizer = AdamW

4. Report of model B

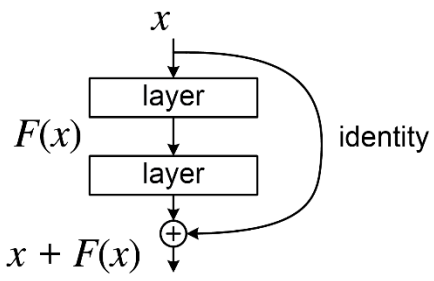
In model B, I used the resnet152 pretrained model, I found that the more deep the resnet model, the better acc I can get. I have tried the resnet18 and resnet50, and found the resnet152 is best. Due to the input default dim is 224, so I add resize(224), and other data augmentation is same as model A.

Training settings:

1. Data augmentation: [Resize(224),RandomCrop, RandomRotation, RandomHorizontalFlip, Normalize]
2. Number of epoch = 20
3. Batch size = 32
4. lr = 0.003
5. loss_function = SoftCrossEntropy
6. optimizer = SGD

The difference between model A & B:

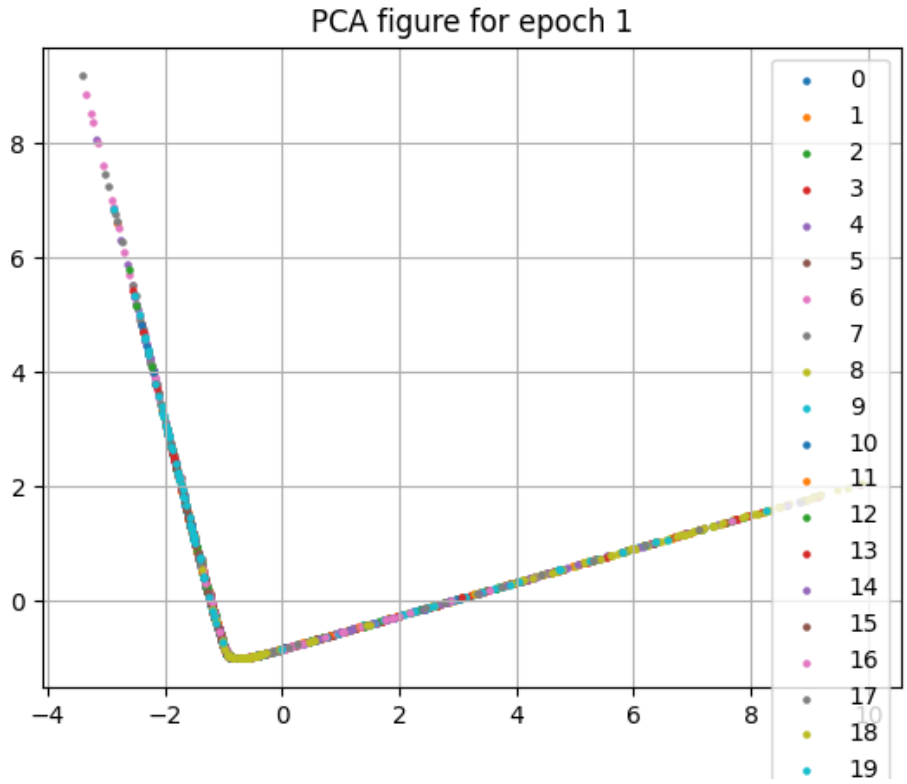
The main difference between model A & B is that the model B can used the pretrained weights, and the resnet model have residual network(adding the output of previous layer to current layer input directly, helping the gradient to backprop and prevent gradient vanishing.)

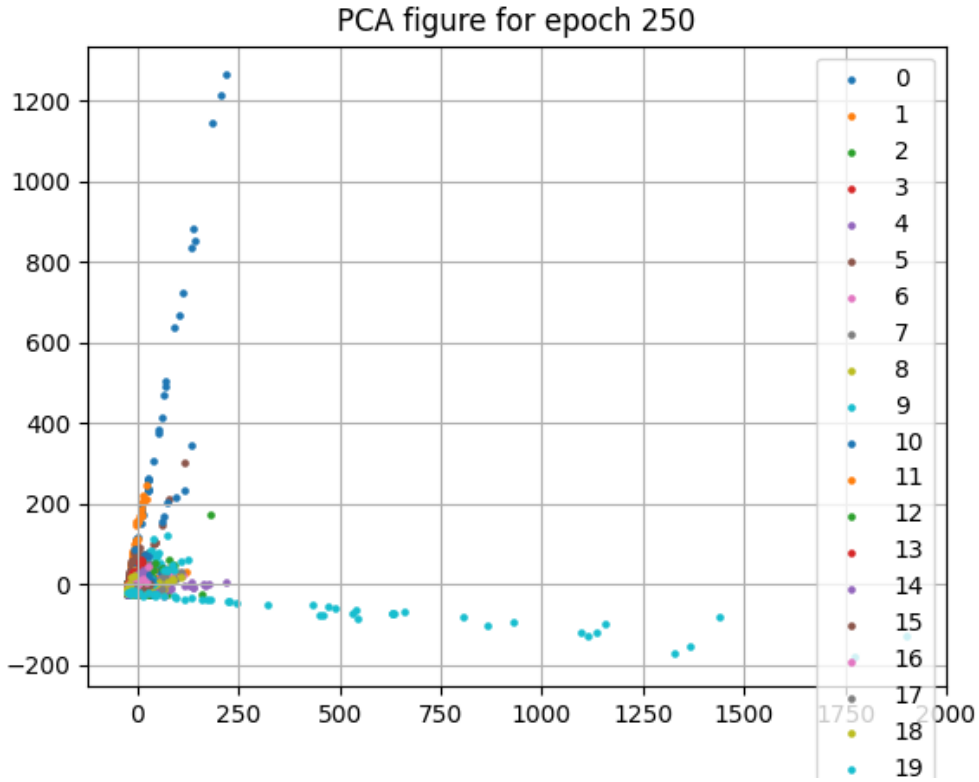


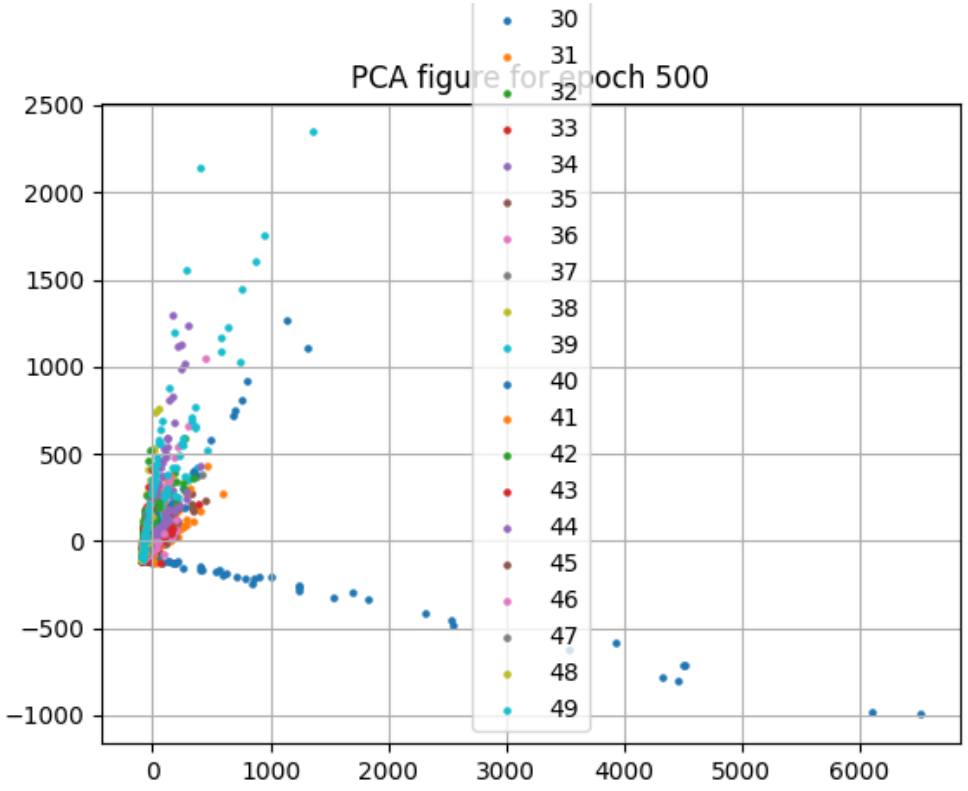
Another difference is that I using the mixup and SoftCrossEntropy in timm,

Mixup: 兩張 img 每個位置像素依比例疊加，label using soft label in accordance with the ratio of two img adding. SoftCrossEntropy is used with Mixup.

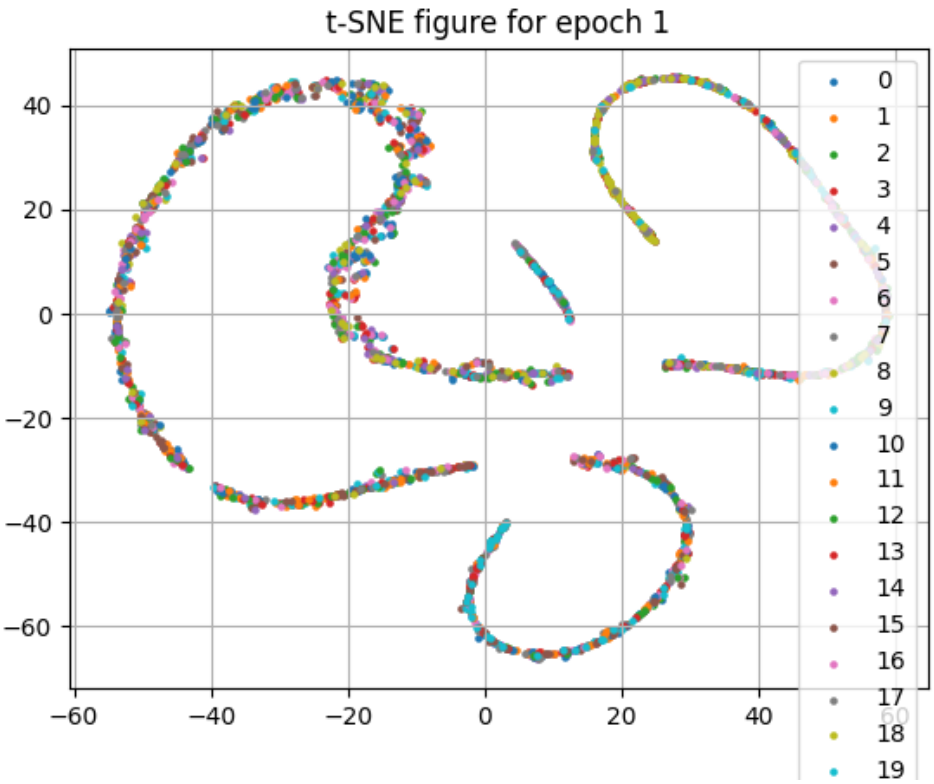
5. Visualize the learned visual representations of model A by PCA
   PCA on second last layer(conv-layer ends output)
   PCA is using a matrix project to low dim, the goal is trying to maintain variance of the data when doing dimension reduction to 2 dim. We can see that the different class are gradually separate after learning, but have very big overlapping it may due to we can't linear project from high dim -> 2 dim very well.

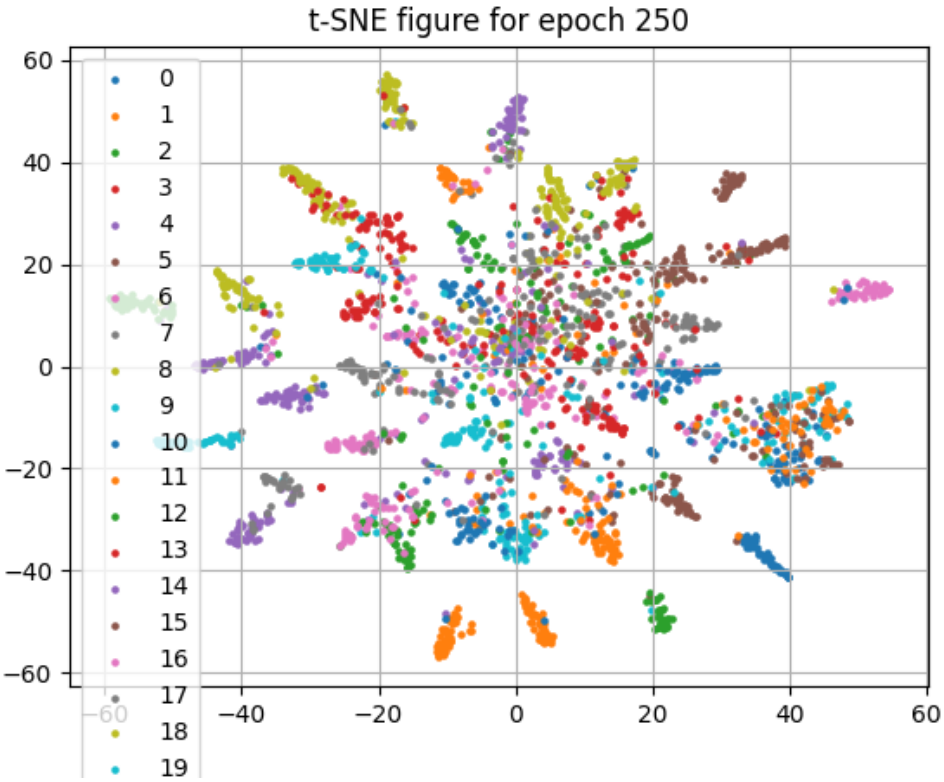


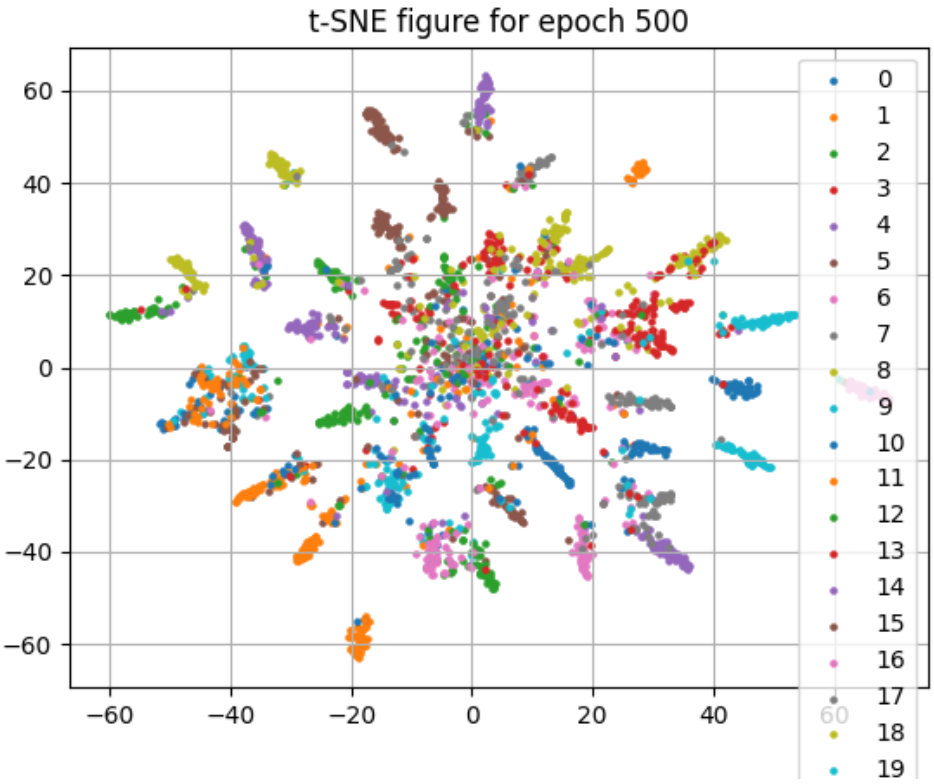

PCA figure for epoch 250
1200
1000
800
600
400
200
0
−200
0
250
500
750
1000
1250
1500
1750
2000
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19


PCA figure for epoch 500
2500
2000
1500
1000
500
0
−500
−1000
0
1000
2000
3000
4000
5000
6000
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

6. Visualize the learned visual representations of model A by t-SNE
   t-SNE on second last layer(conv-layer ends output)
   t-SNE is a unsupervised nonlinear technique that focuses on preserving the pairwise similarities between data points in low-dim. Using the KL-divergence to calculate loss and gradient descent through it.
   Notice that in early stage the model have not train very well so the figure look so mess-up, but in the last epoch the model is train almost acc = 60, so the figure can easily separate the data clearly.




t-SNE figure for epoch 250
60
40
20
0
−20
−40
−60
−60
−40
−20
0
20
40
60
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

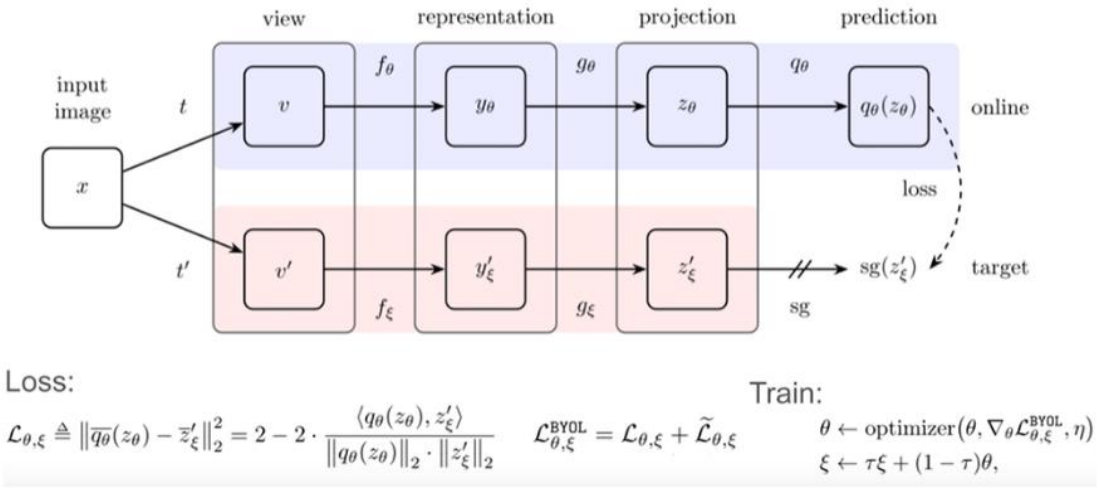

t-SNE figure for epoch 500
60
40
20
0
−20
−40
−60
−60
−40
−20
0
20
40
60
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

Problem 2:Self-supervised pre-training for image classification

1. Implementations details of SSL method for pre-training the ResNet50 backbone

BYOL(Bootstrap your own latent)

BYOL is an unsupervised learning. The figure shows below is its flow-chart, the input image x using two data augmentation t and t' to get two different augmented image, and after feed into the projection, the online network will add another prediction neural network to compute loss(similarity loss) between the target projection output. When optimization, online network using the SGD to gradient descent and the target network using the exponential moving average to gradient descent.



Training settings:

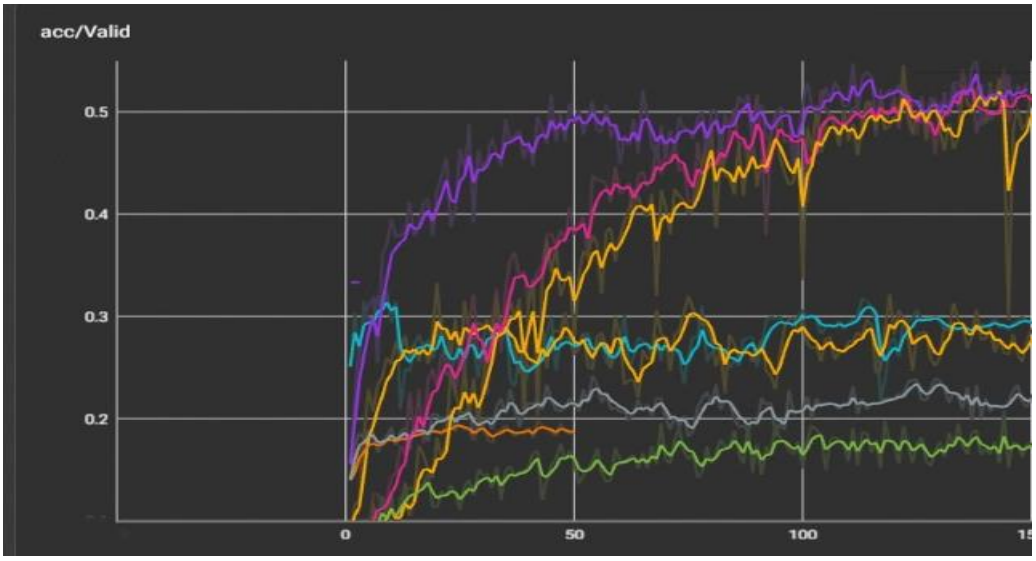
1. Number of epoch = 1000
2. Model = resnet50
3. Img_size = 128
4. Batch size = 128
5. Data Augmentation

```python
tfm = transforms.Compose([
    transforms.RandomApply(
        [transforms.ColorJitter(0.8, 0.8, 0.8, 0.2)],
        p = 0.3
    ),
    transforms.RandomGrayscale(p=0.2),
    transforms.RandomHorizontalFlip(),
    transforms.RandomApply(
        [transforms.GaussianBlur((3, 3), (1.0, 2.0))],
        p = 0.2
    ),
    transforms.RandomResizedCrop((128, 128)),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=torch.tensor([0.4705, 0.4495, 0.4037]),
        std=torch.tensor([0.2170, 0.2149, 0.2145])),
])
```

6. Optimizer = Adam
7. lr = 1e-4
8. scheduler = CosineAnnealingLR

2. Downstream task for Office-Home Image-Classification

Setting	Pre-training(Mini-Image-Net)	Fine-tuning(Office-Home)	Valid Accuracy
A		Train full model	0.5271
B	w/ label(TA backbone)	Train full model	0.5527
C	w/o label(My backbone)	Train full model	0.5505
D	w/ label (TA backbone)	Fix the backbone	0.2422
E	w/o label(My backbone)	Fix the backbone	0.2084



Training settings(Model A to E setting is consistent):

1. Data Augmentation

```
tfm = transforms.Compose([
    transforms.RandomApply(
        [transforms.ColorJitter(0.8, 0.8, 0.8, 0.2)],
        p = 0.3
    ),
    transforms.RandomGrayscale(p=0.2),
    transforms.RandomHorizontalFlip(),
    transforms.RandomApply(
        [transforms.GaussianBlur((3, 3), (1.0, 2.0))],
        p = 0.2
    ),
    transforms.RandomResizedCrop((128, 128)),
    transforms.ToTensor(),
    transforms.Normalize(
        mean=torch.tensor([0.4705, 0.4495, 0.4037]),
        std=torch.tensor([0.2170, 0.2149, 0.2145])),
])
```

2. Batch size = 64
3. Model = resnet50 + Dropout + Linear(1000,65)
4. Number of epoch = 150
5. lr = 5e-4
6. Loss function = CrossEntropy
7. optimizer = Adam

Discuss the results:

Compared setting A to C:

The setting A to C is training the whole network(backbone + classifier), The comparison between result A and result B, C reveals a 3% difference, indicating using pretrained backbone can enhance the accuracy as we expected. The setting B(with label) and setting C(without label) seems didn't have much difference in valid accuracy.

Compared setting D&E:

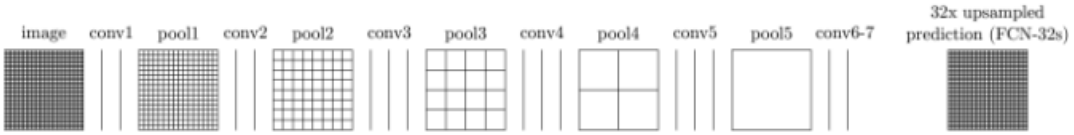
The setting D & E backbone are frozen, only the classifier can optimize, the dataset of pretrain is Mini-Image-net and downstream is Office Home dataset, so it's obvious that the validation accuracy will worse than the setting A to C which can training the whole network(backbone + classifier).Results find that the supervised backbone(TA provided) the valid accuracy is better than the unsupervised backbone(My backbone).

Problem 3:Semantic Segmentation

1. Implementations Details of model A

Model A:

Using the pretrained vgg16 model backbone(CNN layer). Due to the vgg16 have 5 blocks, each blocks using a max-pooling so it decrease the img / 2^5 = img / 32, we add the conv 6-7 and add the upsampled ConvTranspose2d to let output dim same as input img.

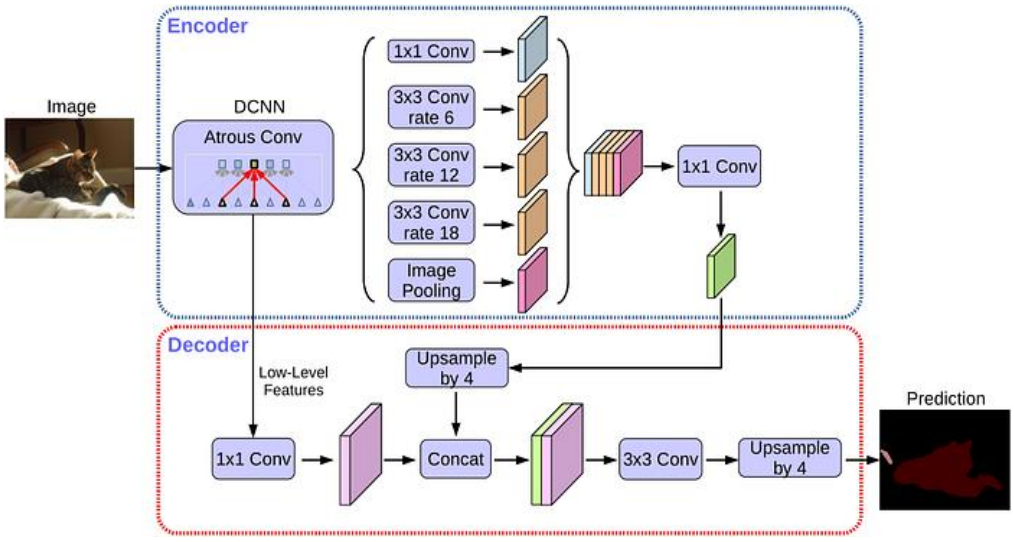


Training settings:

1. Data Augmentation: [ToTensor(), Normalize()] (In trail and error, I found that the HorizontalFlip and others seem no obvious improvement so I drop it)
2. Number of epoch = 50
3. Batch size = 4
4. Loss function = CrossEntropy
5. lr = 0.001

2.Implementations Details of Model B:

I choose the pretrained deeplabv3_resnet50 as my model B.



The difference between DeepLabv3+ and FCN32s is that the DeepLab3+ using

the Atrous Conv. Atrous Conv is using a dilation rate to control your conv kernel to get more local area or more widely. Another difference is using different kernel size conv-layer to get different detail and concat the output through the next layer.

Training settings:

1. Data Augmentation: [ToTensor(), Normalize()]
2. Number of epoch = 80
3. Batch size = 12 (I using the Kaggle to train, due to my laptop vram limited
4. lr = 0.01
5. Loss function = Poly1Loss

   https://arxiv.org/pdf/2204.12511.pdf

   2022 paper, polyloss is using the taylor expansion improving the formula of CrossEntropy and focal loss, I found that the data of segmentation is imbalance, the mask picture sometimes it just whole same color, so I use this new loss function in order to combat the data imbalance.

$$L_{\text{CE}} = -\log(P_t) = \sum_{j=1}^{\infty} 1/j(1-P_t)^j = (1-P_t) + 1/2(1-P_t)^2 \ldots$$

$$L_{\text{FL}} = -(1-P_t)^{\gamma}\log(P_t) = \sum_{j=1}^{\infty} 1/j(1-P_t)^{j+\gamma} = (1-P_t)^{1+\gamma} + 1/2(1-P_t)^{2+\gamma} \ldots$$

The poly1loss formula is below

$$L_{\text{Poly-N}} = \underbrace{(\epsilon_1+1)(1-P_t) + \ldots + (\epsilon_N + 1/N)(1-P_t)^N}_{\text{perturbed by } \epsilon_j} + \underbrace{1/(N+1)(1-P_t)^{N+1} + \ldots}_{\text{same as } L_{\text{CE}}}$$

$$= -\log(P_t) + \sum_{j=1}^{N} \epsilon_j (1-P_t)^j$$

3. mIoU on model A & B

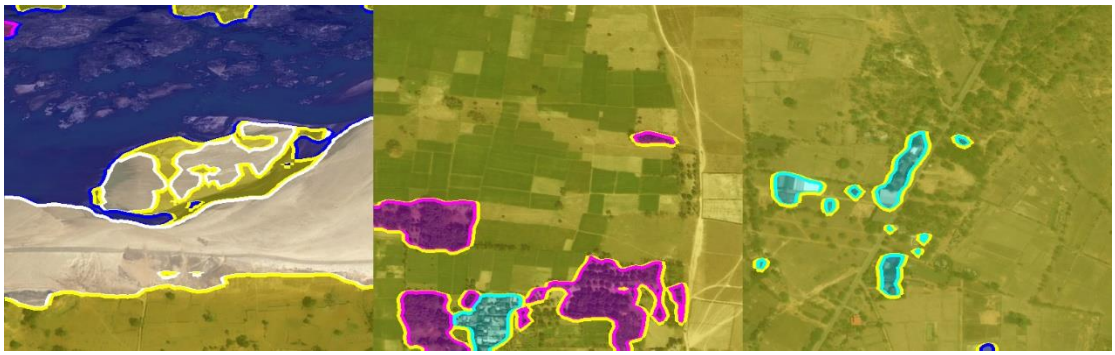
	Model A(FCN32s)	Model B(DeepLabv3+)
mIoU	0.6027	0.7467

4. Show the predicted segmentation mask

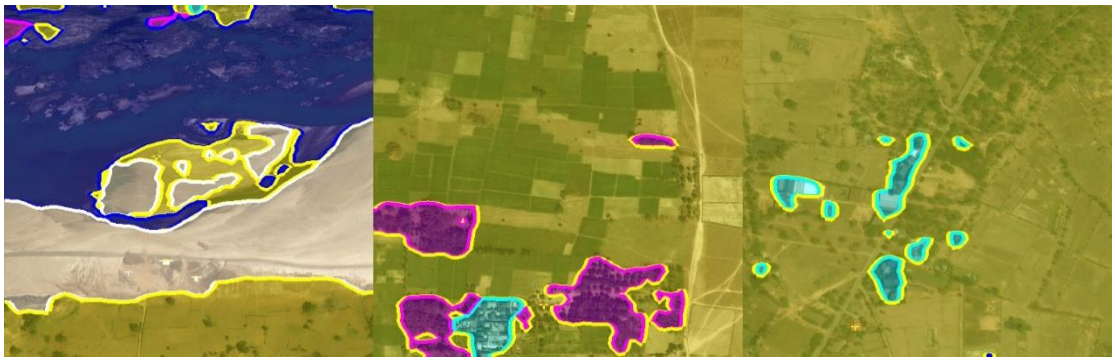
0013_sat.jpg	0062_sat.jpg	0104_sat.jpg

Epoch 1:(Early stage)

Epoch 40:(Middle stage)



Epoch 80:(Final stage)



Ground Truth: