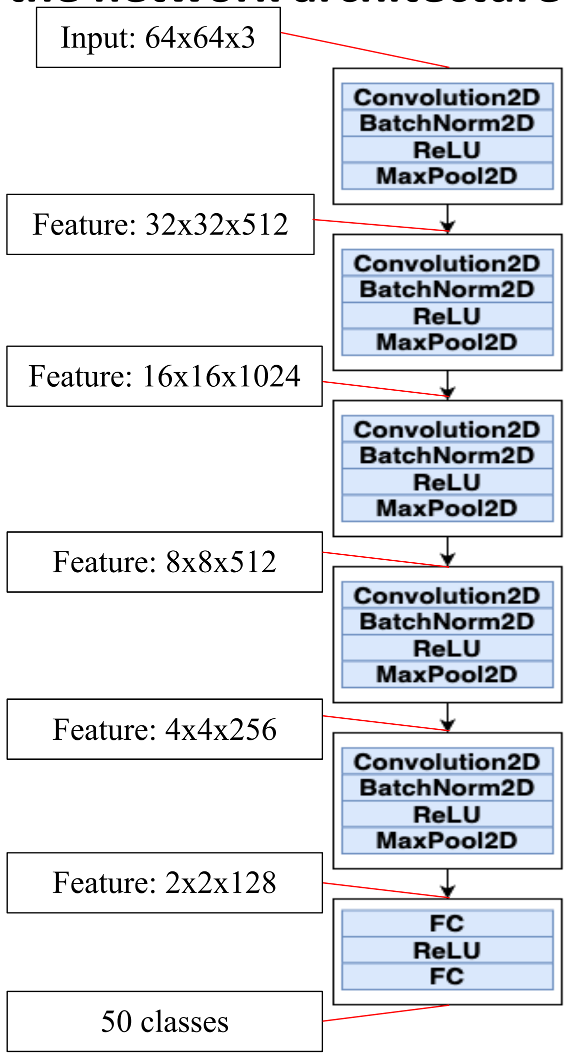
Student ID: R10625016

Name: 許致銓

Department: 森林所

Problem1: Image Classification

1. Draw the network architecture of method A or B.



Model A:

I design a CNN model with 3x3 convolution kernel, batch normalization, ReLU, and max pooling as above by myself. Eventually, use fully connected layer to output the probability for 50 classes.

2. Report accuracy of your models (both A, B) on the validation set.

Model A: 0.69 (Self-designed CNN + classifier)
Model B: 0.9088 (ResNext101 + classifier)

3. Report your implementation details of model A

- Optimizer:
  - AdamW. With lr = 0.0005, weight decay = 0.01
- Loss Function:
  - Cross Entropy
- Cross Validation:
  - I misled the meaning of “not use validation data to train your model”. Hence, in problem 1, I didn’t use validation data as my validation data. Instead, I split the training set with 0.05 percent (22500 x 0.05 = 1125 images) as my validation set. Besides, I use the validation data (val_50) as my testing set. I knew it is a little bit dumb after discussed with my roommate, so I adjusted in problem2&3.
- Learning Rate Scheduling:
  - CosineAnnealingLR, T_max = 10, eta_min = 0.000025
- Batch Size:
  - 64 in training. 32 in inferencing.
- Epochs:
  - 100 with early stop 25.
- Data Augmentation:
  - Resize to (64, 64)
  - Random horizontal flip with p = 0.5
  - Random rotation with 30 degrees.

4. Report your alternative model or methods in model B, and describe its difference from model A.

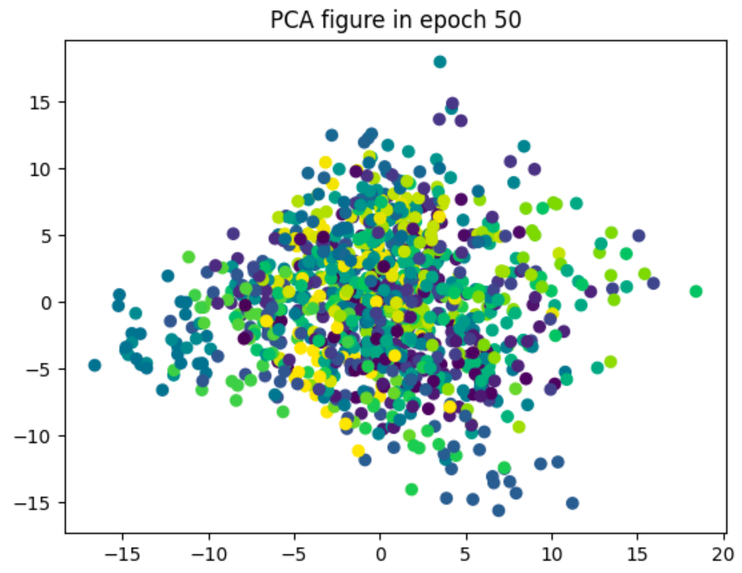
Model B: ResNext101_64x4d_v1 (Selected from the better model on PyTorch website. By the way, I did try lots of models on the website.)

Difference: ResNext101 is a super useful model! In my opinion, it is a better option not just because of its higher parameter numbers, but also due to its "cardinality" (branches, groups). Compare to my self-designed CNN model, it provides more features. Moreover, it converges in 66 epochs, faster than model A in 76 epochs. From my perspectives, it is related to the "shortcut" for easier training.

Implementation Details:

- Optimizer:
  - AdamW with learning rate = 0.000075, weight decay = 0.01
- Loss Function:
  - Cross Entropy
- Cross Validation: Same as the above misleading.
- Learning Rate Scheduling:
  - StepLR with gamma = 0.1, step = 10.
- Batch Size:
  - 64 in training. 32 in inferencing.
- Epochs:
  - 100 with early stop 25.
- Data Augmentation:
  - Resize to (100, 100)
  - Random horizontal flip with p = 0.5
  - Random grayscale with p = 0.2
  - Random rotation with 30 degrees.
  - Gaussian blur with kernel = 3, sigma = (0.1, 2.0)

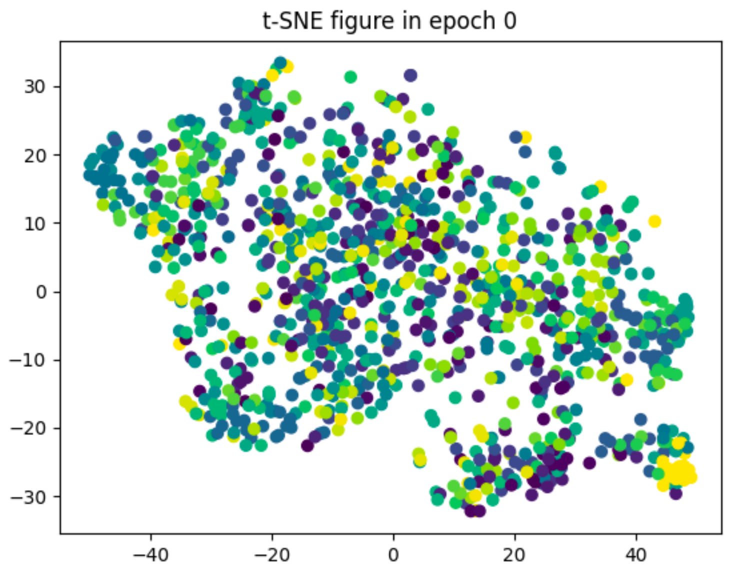
5. **Visualize the learned visual representations of model A on the validation set by implementing PCA (Principal Component Analysis) on the output of the second last layer. Briefly explain your result of the PCA visualization.**

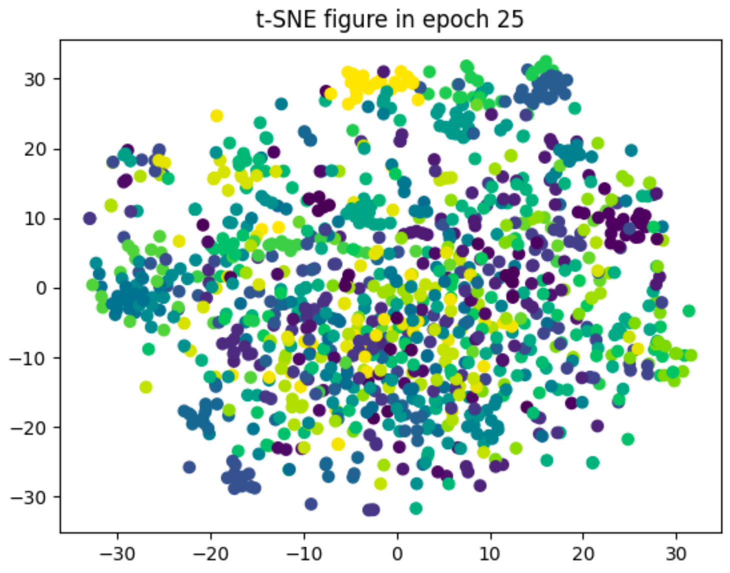


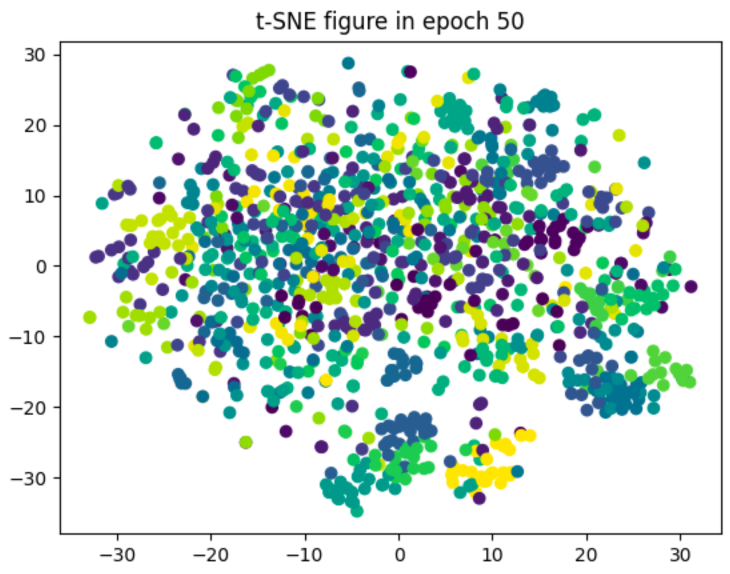
My best model is obtained in epoch 50. However, the feature extracted didn't have a great performance on PCA. In my opinion, this resulted from the number of classes, for 50 classifications to 2 principal components. Without any doubt, it is hard to make the separation of different classes. Perhaps, I need to increase the number of principal components.

6. **Visualize the learned visual representation of model A, again on the output of the second last layer, but using t-SNE (t-distributed Stochastic Neighbor Embedding) instead. Depict your visualization from three different epochs including**

the first one and the last one. Briefly explain the above results.







I found my best model in epoch 50. As you can see, at epoch 0, there are barely clusters in the figure. As in epoch 25, there are still no obvious clusters. In the best epoch 50, you can find 3 clusters in the lower part of the figure. Although it is not apparent, you can see the difference after training.

Problem2: Self-supervised pre-training for image classification

1. **Describe the implementation details of your SSL method for pre-training the ResNet50 backbone. (Include but not limited to the name of the SSL method you used, data augmentation for SSL, learning rate schedule, optimizer, and batch size setting for this pre-training phase)**

   SSL Method: BYOL (Bootstrap Your Own Latent)

   - Optimizer:

    - AdamW with learning rate = 0.0001, weight decay = 0.0005
- Learning Rate Scheduling:
    - CosineAnnealingLR, T_max = 10, eta_min = 0.000005
- Hidden Layer: Average Pooling
- Batch Size: 128
- Epochs: 1000
- Data Augmentation:
    - Resize to (128, 128)
    - Random horizontal flip with p = 0.5
    - Normalization with mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]

2. Please conduct the Image classification on Office-Home dataset as the downstream task. Also, please complete the following Table, which contains different image classification setting, and discuss/analyze the results.

Setting	**Pre-training (Mini-ImageNet)**	**Fine-tuning (Office-Home dataset)**	**Validation accuracy (Office-Home dataset)**
A	-	Train full model (backbone + classifier)	*0.172*
B	w/ label (TAs have provided this backbone)	Train full model (backbone + classifier)	0.379
C	w/o label (Your SSL pre-trained backbone)	Train full model (backbone + classifier)	*0.461*
D	w/ label (TAs have provided this backbone)	Fix the backbone. Train classifier only	*0.195*
E	w/o label (Your SSL pre-trained backbone)	Fix the backbone. Train classifier only	*0.308*

Implementation details

- Optimizer:

    - Adam with learning rate = 0.0003, weight decay = 0.0005
- Loss Function:
    - Cross Entropy
- Learning Rate Scheduling:
    - StepLR with gamma = 0.1, step = 10
- Batch Size:
    - 64
- Epochs:
    - 200 with early stop 50
- Data Augmentation:
    - Resize to (128, 128)
    - Random horizontal flip with p = 0.5
    - Random grayscale with p = 0.2
    - Random rotation with 30 degrees.
    - Normalization with mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]

In setting A, I cannot get a great result and it is reasonable. In setting B, with the semi-supervised learning, it increases to meet the simple baseline 0.36. Later in setting C, with my BYOL pretext (backbone: resnet50), it attained 0.46 higher than strong baseline. Obviously, it claimed that self-supervised training in BYOL is helpful and even beneficial than semi-supervised learning! In setting D, I cannot train the model for a good performance. I think it is because that my classifier consists of only 3 layers, which is too shallow. Lastly, in setting E with the same shallow classifier, pretext outweighs the semi-supervised learning again!
From my perspective of view, my pretext is strong enough for learning the features of the un-supervised task and it may attribute to the long training time with 1000 epochs.

Problem3: Semantic Segmentation

1. (3%) Draw the network architecture of your VGG16-FCN32s model (model A).

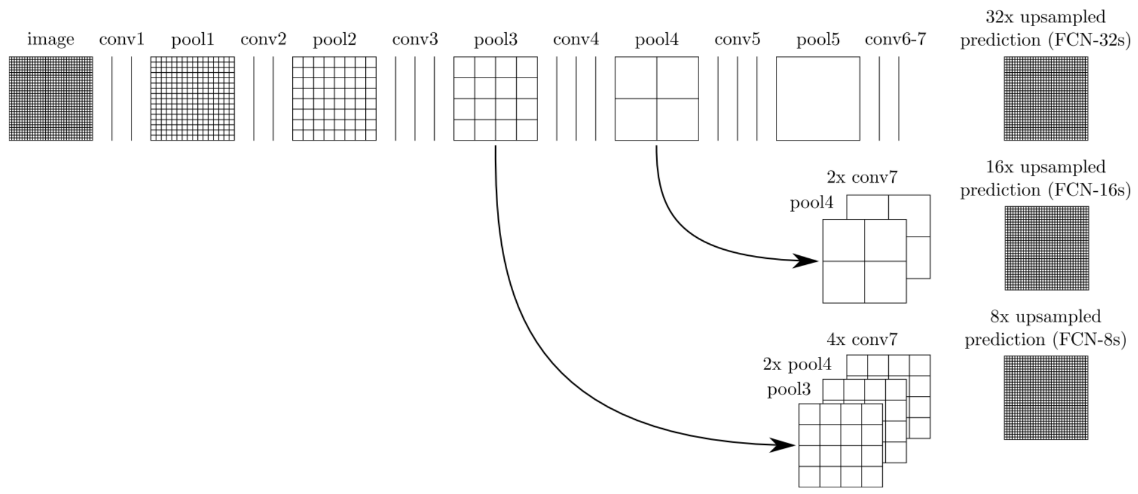


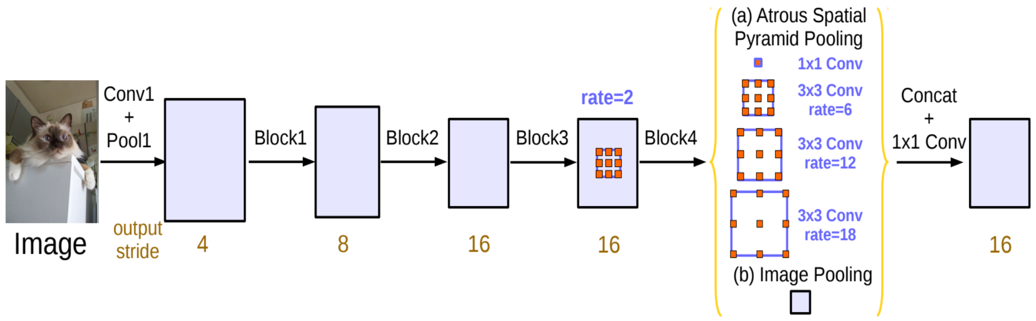
image
conv1
pool1
conv2
pool2
conv3
pool3
conv4
pool4
conv5
pool5
conv6-7
32x upsampled prediction (FCN-32s)
2x conv7
pool4
16x upsampled prediction (FCN-16s)
4x conv7
2x pool4
pool3
8x upsampled prediction (FCN-8s)

(from "Fully Convolutional Networks for Semantic Segmentation")

I use FCN32 architecture with 5 transpose convolution layers, each with 2x upsample. Moreover, between 2 transpose convolutional layers, there is a batch normalization layer.

2. (3%) Draw the network architecture of the improved model (model B) and explain it differs from your VGG16-FCN32s model.

Model B: DeepLabv3 with backbone ResNet50, weights = COCO_WITH_VOC_Labels_V1.
I found this stronger model on the same website as the above but related to semantic segmentation.
Why not ResNet101? Because of memory limitation.



(from “Rethinking Atrous Convolution for Semantic Image Segmentation”)

“Atrous” means “holes” in French, and it means dilated convolution in deep learning. The main idea in this work is that they consider features in different scale by dilated kernels. As you can see, in Atrous Spatial Pyramid Pooling layer (a), they provide both fine-grained and coarse-grained features. In this way, it can predict better than FCN32, and it is totally different from the previous ideas to extract different scale features (original pyramid concept).

In my opinion, if I choose to add some U-Net ideas (shortcut) into FCN32, I may achieve some great performance. However, it may result in more memory costs compared to DeepLabv3.

3. (1%) Report mIoUs of two models on the validation set.

Model A (VGG16 + FCN32): 0.671 with 60 epochs
Model B (DeepLabv3): 0.741 with 15 epochs

4. (3%) Show the predicted segmentation mask of “validation/0013_sat.jpg”, “validation/0062_sat.jpg”, “validation/0104_sat.jpg” during the early, middle,

and the final stage during the training process of the improved model.

- **Tips: Given n epochs training, you could save the 1st, (n/2)-th, n-th epoch model, and draw the predicted mask by loading these saved models.**

sat.jpg (13, 62, 104)	Ground Truth	Epoch 0	Epoch 7	Epoch 14