Deep Learning and Computer Vision

HW1 Report

Name: 謝銘倫 **ID: B10502166**

1 Problem_1

1.1 Implementation Details of the Self-Supervised Learning Method

I integrated BYOL (Fig.1) into my neural network as a self-supervised learning approach. Additionally, I applied data augmentation techniques, such as resizing, random horizontal flipping, and color jittering, to mitigate overfitting caused by model complexity.

To optimize the learning process, I employed the SGD optimizer along with the ReduceLROnPlateau learning rate scheduler. The advantage of using SGD is its ability to update parameters using only a small batch of data at each iteration, making it much faster per iteration compared to batch gradient descent. While ReduceLROnPlateau allows for dynamic adjustment of the learning rate based on validation performance.

Through experimentation, I found that setting the batch size to 16 and the learning rate to 0.005 yielded the best results.

The data augmentation details are outlined as follows:

```
from torchvision import transforms

transform=transforms.Compose(
        [
            transforms.Resize((128, 128)),
            transforms.RandomResizedCrop(128, scale=(0.08, 1.0)),
            transforms.RandomHorizontalFlip(),
            autoaugment.RandAugment(num_ops=2, magnitude=9),
            transforms.ColorJitter(
                brightness=0.4, contrast=0.4, saturation=0.4, hue=0.1
            ),
            transforms.RandomRotation(20),
            transforms.ToTensor(),
            transforms.Normalize(
                mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]
            ),
            RandomErasing(
                p=0.5, scale=(0.02, 0.33), ratio=(0.3, 3.3), value=0, inplace=
                    False
            ),
        ]
```

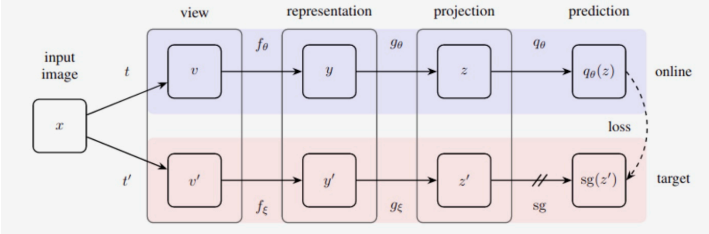


Figure 1: BYOL Training Process

1.2 Mean classification accuracy on validation set.

Setting	Pre-training (Mini-ImageNet)	Fine-tuning (Office-Home dataset)	Validation accuracy (Office-Home dataset)
A	-	Train full model (backbone + classifier)	0.4409
B	w/ label (TAs have provided this backbone)	Train full model (backbone + classifier)	0.4778
C	w/o label (Your SSL pre-trained backbone)	Train full model (backbone + classifier)	0.4803
D	w/ label (TAs have provided this backbone)	Fix the backbone. Train classifier only	0.1650
E	w/o label (Your SSL pre-trained backbone)	Fix the backbone. Train classifier only	0.0911

Table 1: Pre-training and Fine-tuning Settings with Validation Accuracy

Analysis:

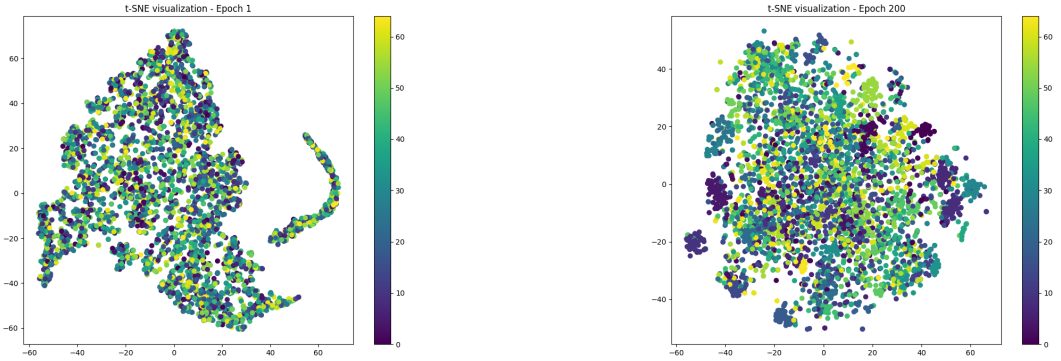
1. **Full model fine-tuning is crucial:** Fixing the backbone and only training the classifier (Settings D, E) dramatically reduces performance. Emphasizes the importance of adapting the entire model to the new dataset

2. **SSL effectiveness:** Self-supervised learning (Setting C) slightly outperforms supervised pre-training (Setting B) when fine-tuning the full model. Demonstrates the potential of SSL in transfer learning scenarios

Training Settings:

- number of epochs = 200
- Learning rate = 0.0001
- Batch size = 16
- Optimizer: AdamW
- Criterion: CrossEntropyLoss
- Scheduler = CosineAnnealingLR

1.3 Visualize t-SNE



(a) Epoch = 1 (b) Epoch = 200

Figure 2: First and last epoch of t-SNE Comparison

Analysis:

1. **Epoch 1 (Fig.a)**
   - **Scattered Distribution:** The points are widely spread out with no clear clustering.
   - **Mixed Colors:** There's a high degree of color mixing throughout the plot, indicating that different classes (represented by colors) are not well-separated.

2. **Epoch 200 (Fig.b)**
   - **Distinct Clusters:** Multiple clear clusters have formed, indicating the model has learned to group similar data points together.
   - **Color Separation:** There are now visible regions of similar colors, showing that the model has learned to separate different classes.

2 Problem_2

2.1 VGG16-FCN32s

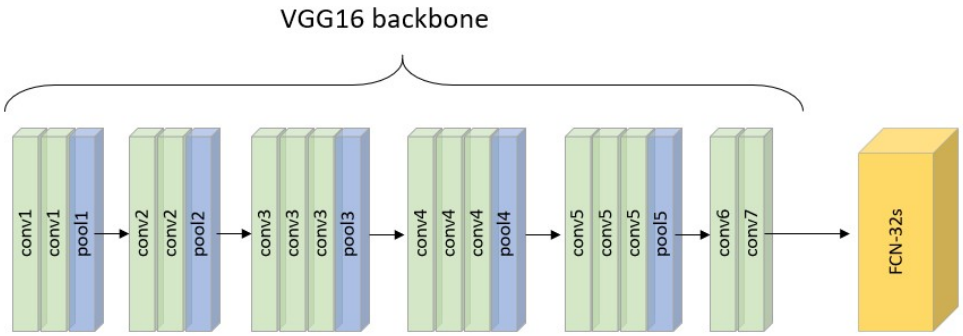


Figure 3: VGG16-FCN32 Model Architecture

2.2 DeepLabv3

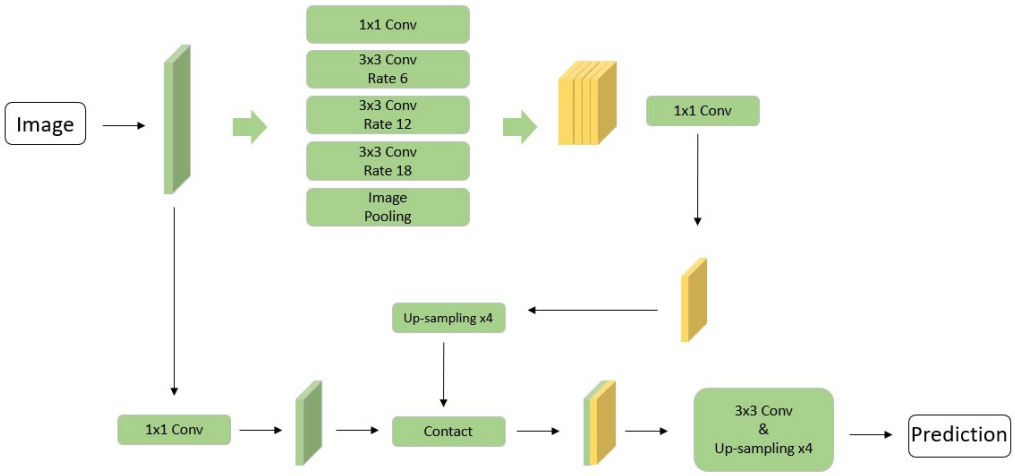


Figure 4: DeepLabv3 Model Architecture

DeepLabv3 enhances semantic segmentation capabilities through its sophisticated use of atrous convolutions and ASPP, allowing it to perform better on images with varying object sizes and complex backgrounds compared to the more straightforward VGG16-FCN32 architecture. While VGG16-FCN32 is simpler and faster, it often struggles with detailed segmentation tasks due to its pooling layers and fixed structure.

2.3 Report Performance

VGG16-FCN32 miou: 0.0634
Deeplabv3 miou: 0.7445

2.4 Show predicted segmentation mask

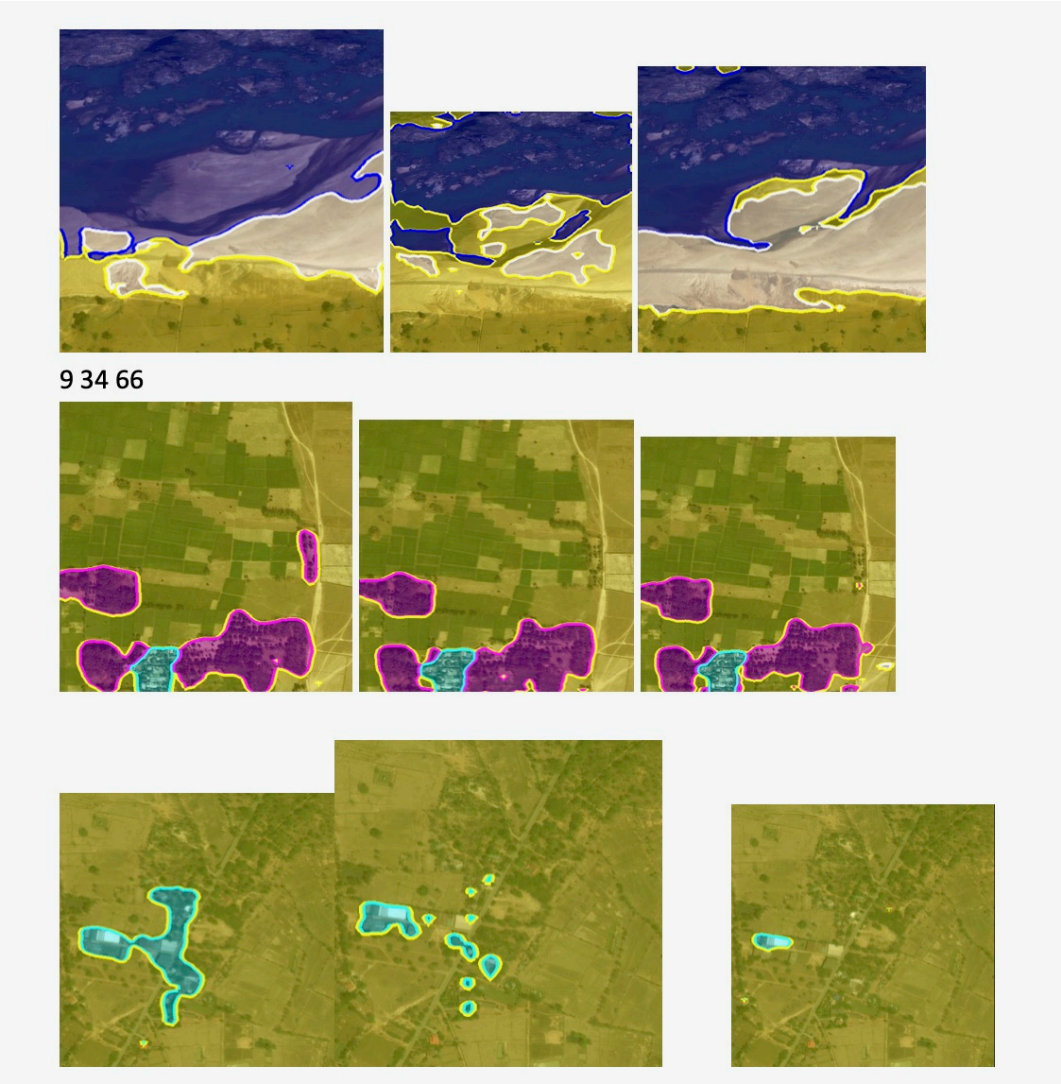


Figure 5: Segmentation mask of 0013_sat, 0062_sat, 0104_sat from top to bottom, for each epoch = 9, epoch = 34, and epoch = 66 from left to right.

mIou

epoch = 9: 0.63856

epoch = 34: 0.58695

epoch = 66: 0.72501

2.5 SAM

I use official website of SAM demo to implement SAM.(`https://segment-anything.com/`)

There are three results:

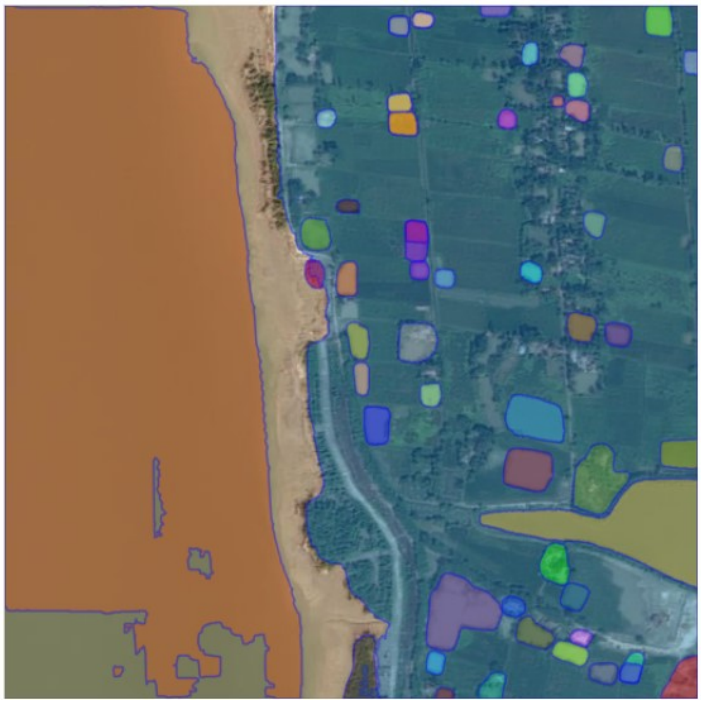


Figure 6: 0211_sat.jpg

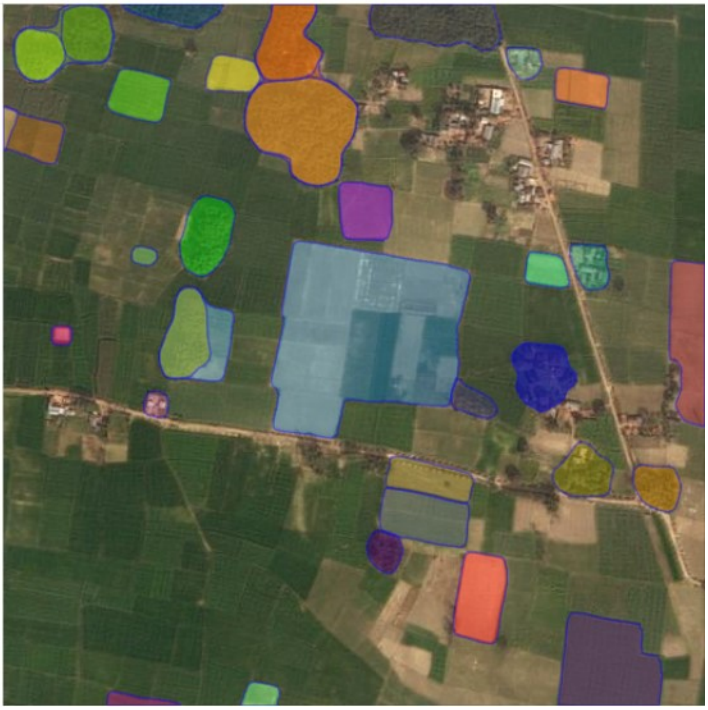


Figure 7: 0111_sat.jpg

Figure 8: 0029_sat.jpg

Reference:

1. **Collaborator:** 楊曄昕 ID: B10508026
2. **HW1 Slide** link
3. **BYOL**link
4. **DeepLabv3 —Atrous Convolution (Semantic Segmentation)** link