

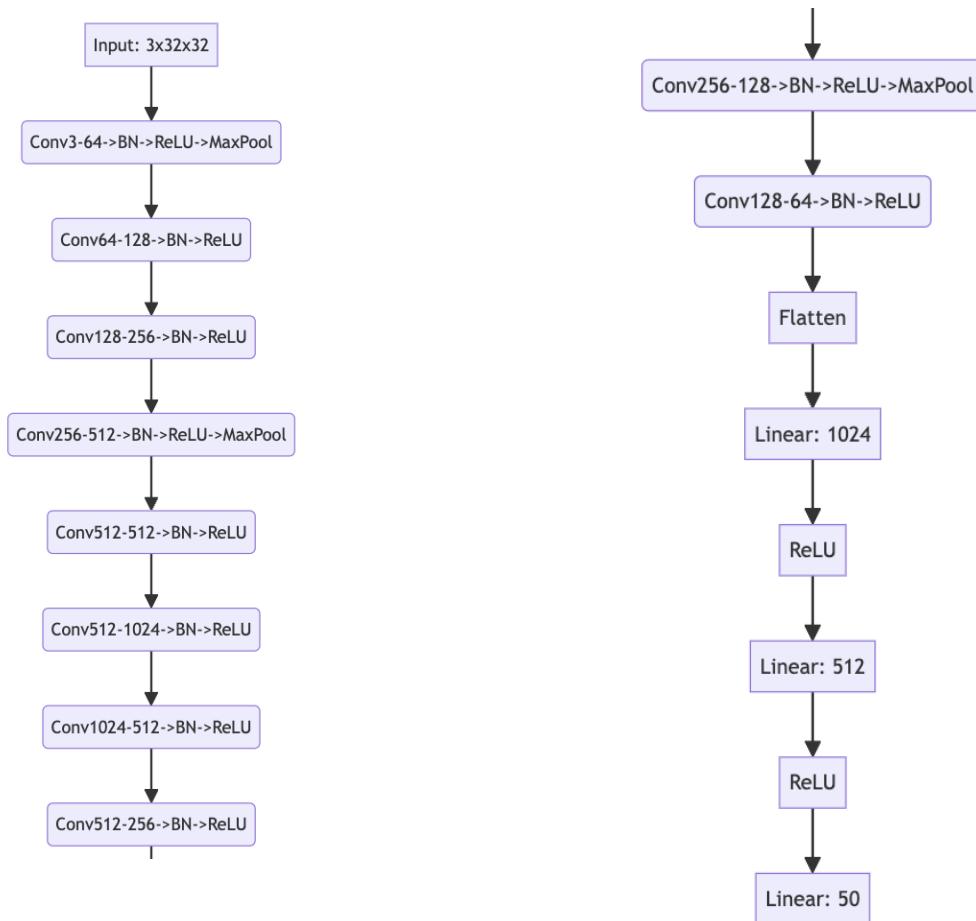
# DLCV-Hw1 Report

R11942180 電信丙碩二 黃湛元

## Problem 1: Image Classification

1. (2%) Draw the network architecture of method A or B.

**Model A architecture :**



2. (1%) Report accuracy of your models (both A, B) on the validation set.

- a. Model A accuracy : 62.56 % (CNN)
- b. Model B accuracy : 90.04 % (Resnext101\_64x4d)

### 3. (2%) Report your implementation details of model A.

Model A 的主要架構是基於 CNN，其中包含多層卷積層、批量正規化層、ReLU 激活函數以及最大池化層。在卷積神經網路後，我們使用了三層全連接層進行分類。訓練參數如下：

- Batch size : 128
- Epoch : 150
- Patience : 15
- Optimizer : AdamW (weight\_decay : 1e-5)
- Learning rate : 3e-4
- Loss function : Cross Entropy Loss
- Scheduler : StepLR(optimizer, step\_size=10, gamma=0.1)
- Cross validation : 我在每一個 training epoch 都使用 validation set 進行驗證，並計算 accuracy。

### 4. (3%) Report your alternative model or method in B, and describe its difference from model A.

我使用 Resnext101\_64x4d 做為 method B 的 model，以下是差異的說明：

#### **ResNeXt101\_64x4d :**

1. **基本結構:** ResNeXt 是 ResNet 的擴充，它結合了 ResNet 和 Inception 的思想。在 ResNeXt 中，每一個 block 不再是單一路徑，而是多個路徑的組合，每條路徑處理不同的 transformations。
2. **Cardinality:** 這是 ResNeXt 的一個主要特性。它指的是 block 內 transformation 的數量。在 **ResNeXt101\_64x4d** 中，cardinality 是 64，意味著每個 block 有 64 條路徑。
3. **深度:** **ResNeXt101\_64x4d** 有 101 層。這使得它在處理複雜的 computer vision task 時，能夠捕捉到更多的特徵。

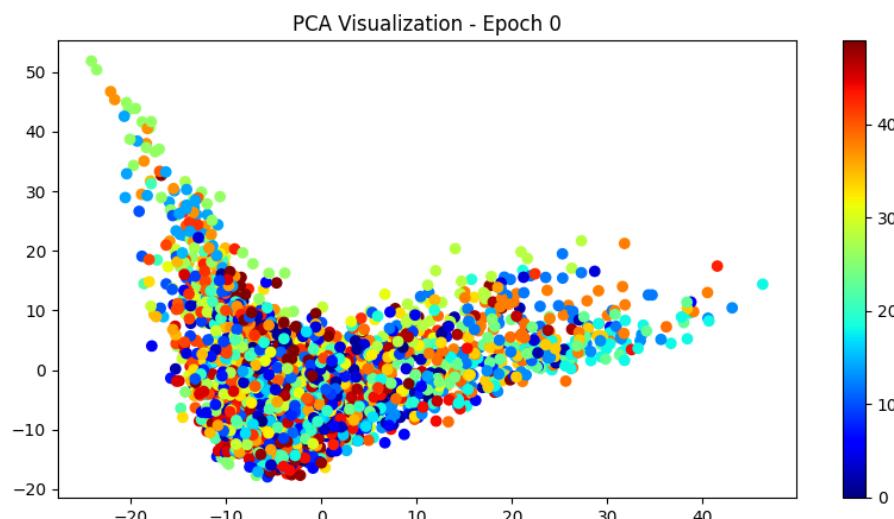
#### **My CNN :**

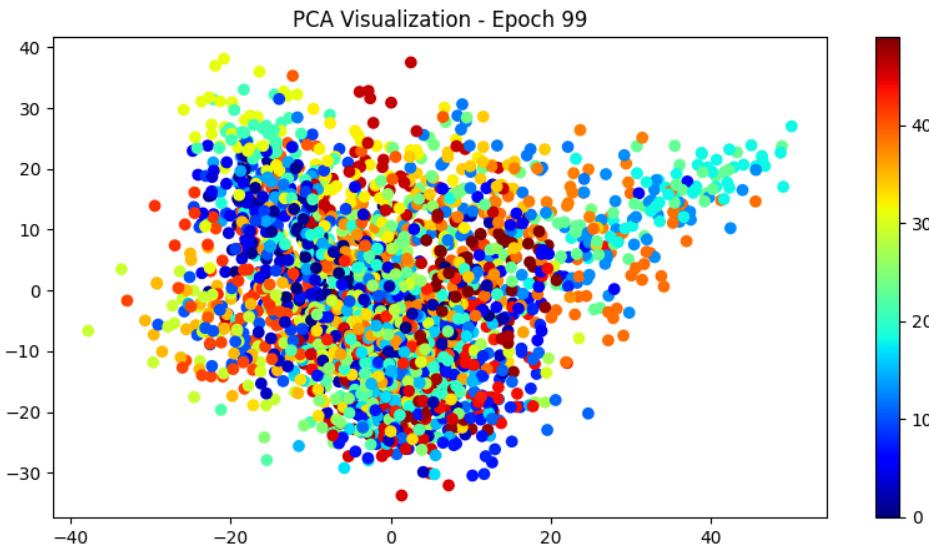
1. **基本結構:** 有多個卷積層、批量標準化層、ReLU，和最大池化層。
2. **深度:** 相較於 **ResNeXt101\_64x4d** 深度較淺。

3. **無殘差結構**: 沒有使用殘差結構，這是 ResNet 和 ResNeXt 的主要特性。

#### 主要差異：

1. **架構深度**: ResNeXt101\_64x4d 較深，這使得它能夠學習更複雜的特徵。
2. **殘差結構**: ResNeXt 使用殘差結構，而我的 CNN 則沒有。這使得 ResNeXt 在訓練深度模型時更加穩定。
3. **多路徑設計**: ResNeXt 使用多路徑設計 (cardinality) 來增強其表現，而我的模型則是單一路徑。
4. **計算量和模型大小**: 由於其深度和多路徑設計，ResNeXt101\_64x4d 的計算量和模型大小都比我的 CNN 要大。
5. **(3%) 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.



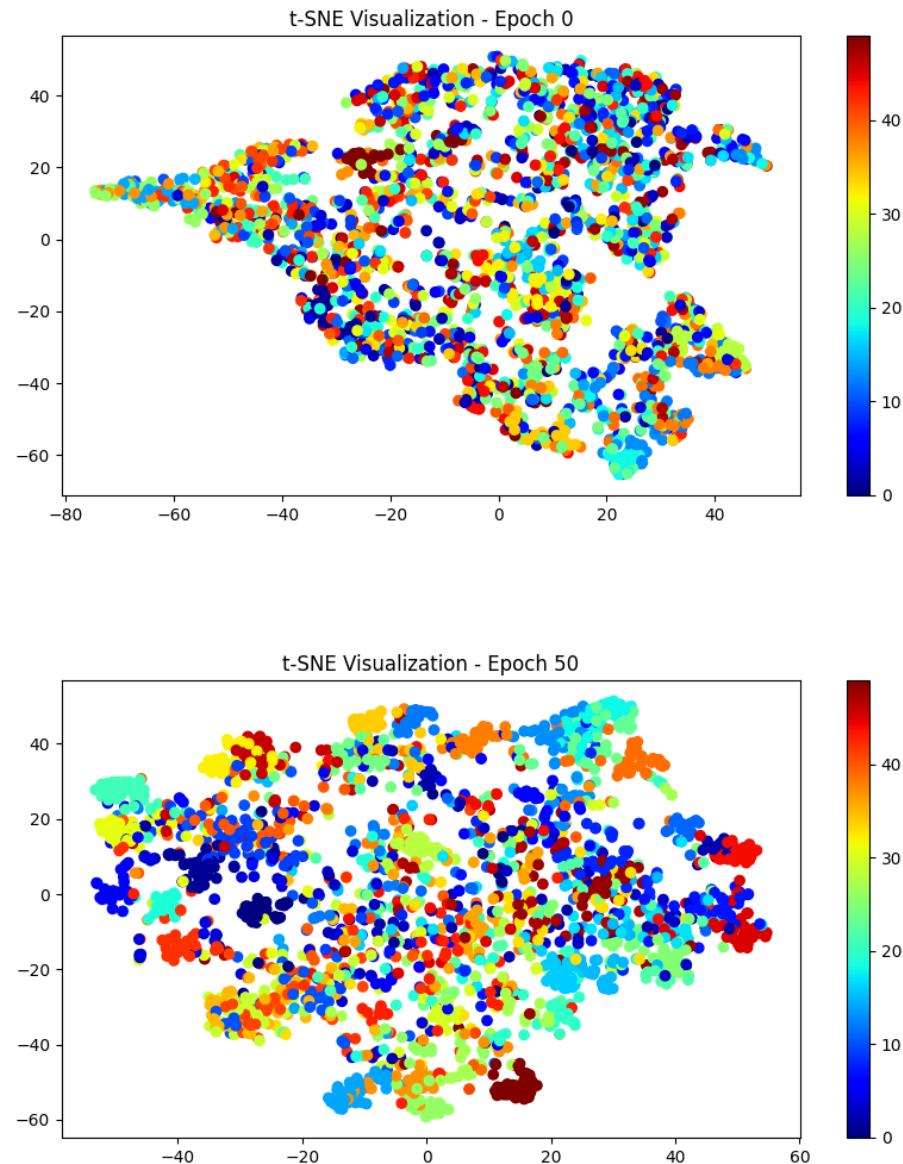


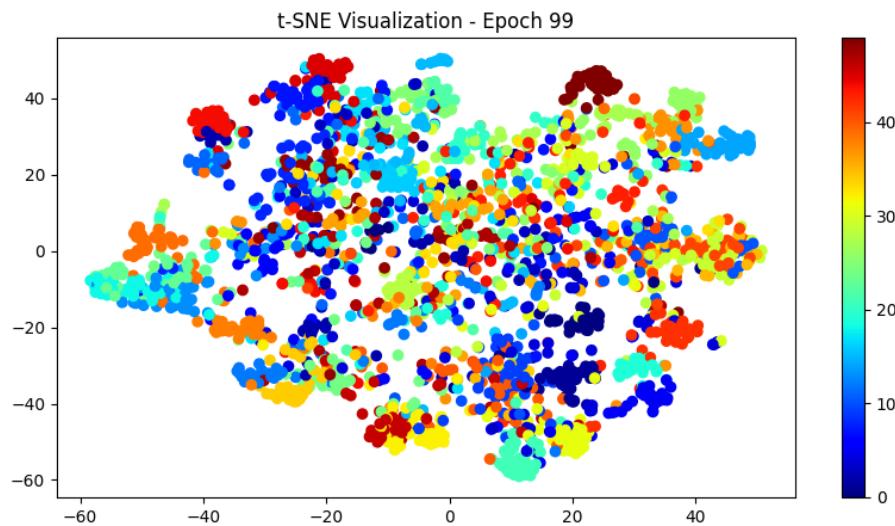
PCA 的目的是找出數據中的主要變異方向，並將其投影到這些方向上以得到低維表示。在這個圖片中，我們可以看到數據沿著某些特定方向有一定的擴展，這些方向即是 PCA 找到的主要變異方向或主成分。

### 1. Epoch 0 vs. Epoch 99:

- **Epoch 0:** 在模型訓練的初期，數據點在 PCA 可視化中呈現一個明確的趨勢，幾乎形成一個彎曲的結構。這可能表示在模型未經訓練時，特徵在某些方向上有一定的分布偏好。
  - **Epoch 99:** 經過許多次的訓練迭代後，數據點在PCA空間中的分布似乎更為均勻且密集。這表明模型在訓練過程中學到了更多的特徵表示，使得各類別間的邊界更加模糊。
2. **分布變化:** 比較兩張圖片，我們可以看到模型從Epoch 0到Epoch 99在特徵空間上的明顯變化。這可能意味著模型已經學習並內化了訓練集中的某些模式，從而更好地區分不同的類別。
  3. **數據分布:** 數據點的分布似乎相對集中，而且沒有形成非常明確的集群。這可能意味著，相對於 t-SNE，PCA 沒有將數據的特徵差異放大，因此在二維空間中較難看出類別間的明確分隔。
  6. **(4%)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.

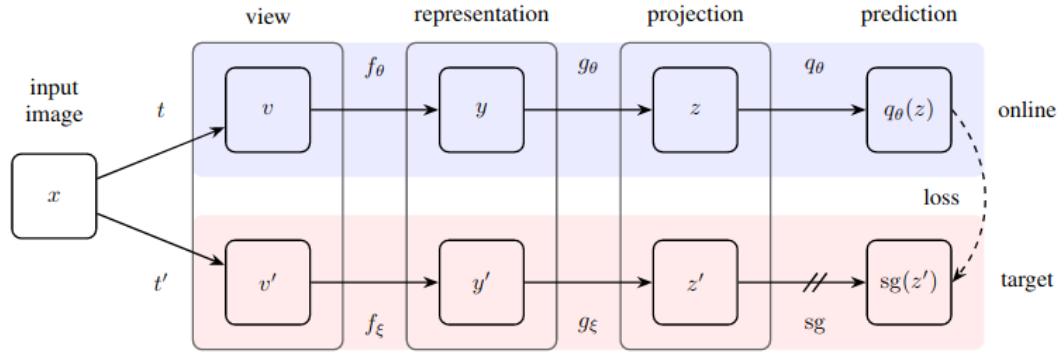




- Epoch 1:** 在訓練的開始階段，數據點似乎較為密集和混雜。這表示模型還未學到明顯的特徵分隔，所以多數的數據點在特徵空間中都是相近的。
- Epoch 51:** 當訓練進行到第50次時，我們可以看到數據點開始形成一些集群，但仍然存在一些混雜的區域。這顯示模型已經開始學習一些區分數據的特徵，但仍需進一步的優化。
- Epoch 100:** 在訓練的最後階段，數據點的集群更加明確，且集群之間的間距也變大。這意味著模型在特徵空間中成功地區分了不同的數據類別。

## Problem 2: Self-Supervised Pre-training for Image Classification

1. (5%) 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)
  - a. 我是使用 BYOL pipeline 來進行 SSL 的 pre-training，示圖如下：



Reference : <https://github.com/lucidrains/byol-pytorch>

b. Training hyperparameter :

- Model : Resnet50
- Batch size : 256
- Epoch : 1000
- Optimizer : AdamW (weight\_decay : 1e-5)
- Learning rate : 1e-4
- Loss function : Cross Entropy Loss
- Scheduler : StepLR(optimizer, step\_size=200, gamma=0.1)
- Cross validation : 我將 training data 以 [0.9, 0.1] 的比例再切出 validation data，並在每一個 training epoch 都使用 validation set 進行驗證和計算 loss。
- Data augmentation : BYOL 預設的 augmentation

```

DEFAULT_AUG = torch.nn.Sequential(
    RandomApply(
        T.ColorJitter(0.8, 0.8, 0.8, 0.2),
        p = 0.3
    ),
    T.RandomGrayscale(p=0.2),
    T.RandomHorizontalFlip(),
    RandomApply(
        T.GaussianBlur((3, 3), (1.0, 2.0)),
        p = 0.2
    ),
    T.RandomResizedCrop((image_size, image_size)),
    T.Normalize(
        mean=torch.tensor([0.485, 0.456, 0.406]),
        std=torch.tensor([0.229, 0.224, 0.225]))
)

```

2. (20%) 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)	25.76 %
B	w/ label (TAs have provided this backbone)	Train full model (backbone + classifier)	45.17 %
C	w/o label (Your SSL pre-trained backbone)	Train full model (backbone + classifier)	41.89 %
D	w/ label (TAs have provided this backbone)	Fix the backbone. Train classifier only	23.15 %
E	w/o label (Your SSL pre-trained backbone)	Fix the backbone. Train classifier only	31.09 %

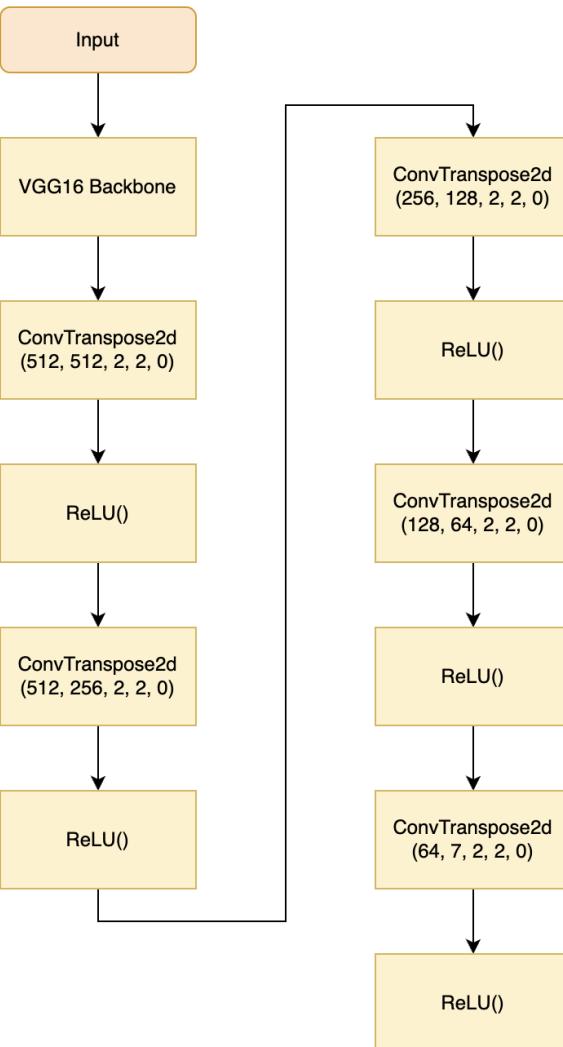
1. **Pre-training 的效果**：從 setting A 和 setting B/C 之間的比較中，我們可以看到 pre-trained model 在 Office-Home dataset 上的效果。當模型沒有在 Mini-

ImageNet 上進行 pre-training 時 (setting A) , 它在 Office-Home 數據集上的準確率只有 25.76%。而當使用有標籤或無標籤的預訓練模型時 (設定 B 和 C) , 其準確率分別提高到 45.17% 和 41.89%。顯示了預訓練的幫助。

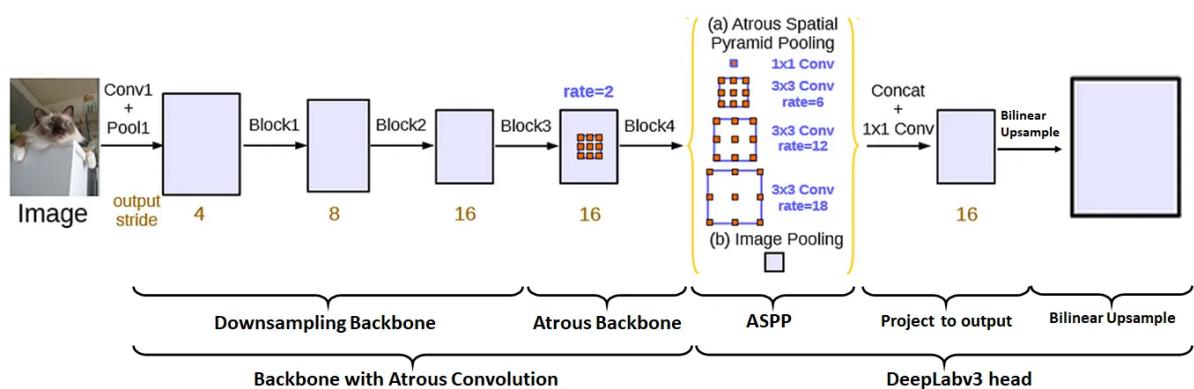
2. **有 label vs. 無 label 的 pre-training**：從 setting B 和 C 之間的比較，我們可以看到有 label 的 pre-trained model (45.17%) 稍微優於無 label 的 pre-trained model (41.89%)。這可能代表有 label 的資料提供了更多的資訊，幫助模型學習更好的特徵表示。
3. **完整模型 vs. 只訓練 classifier**：從設定 B/D 和 C/E 之間的比較，當只訓練分類器並固定 backbone 時，我們可以看到準確率有所下降。這可能是因為 backbone 沒有得到微調的機會，使其適應 Office-Home dataset 的特定特徵。
4. **Pre-training 的質量**：從設定 D 和 E 之間的比較，即使在只訓練分類器的情況下，無 label 的 SSL pre-trained model (31.09%) 仍然優於完全沒有進行 pre-training 的模型 (設定 A 的 25.76%)。這再次強調了 pre-training 的重要性。

### Problem 3: Semantic Segmentation

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



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



Reference : <https://medium.com/@itberrios6/deeplabv3-c0c8c93d25a4#:~:text=Avagliano%20on%20Unsplash-,Introduction,the%20total%20number%20of%20parameters.>

### 1. Backbone:

- **VGG16\_FCN32s**: 使用 VGG-16 作為 feature extractor。VGG-16 是一個深度為 16 層的卷積神經網路，主要由連續的卷積層和池化層組成。VGG 通常有較大的 kernel (例如3x3) 和較小的 step size。
- **DeepLabv3 (ResNet-50)**: 使用 ResNet-50 作為特徵提取器。ResNet-50 是一個深度為 50 層的卷積神經網路，其主要特點是使用殘差塊 (或稱為 skip connections) 以允許更深的網路訓練而不會遇到梯度消失問題。

### 2. Decoder:

- **VGG16\_FCN32s**: 你的模型使用了一系列的逆卷積 (`convTranspose2d`) 層來上採樣特徵圖並將其恢復到原始尺寸。這種上採樣方法比較簡單。
- **DeepLabv3 (ResNet-50)**: DeepLabv3 使用了 atrous 空洞卷積的技術，這使得網路可以採集 multi-scale 的資訊。此外，它還使用了 ASPP (Atrous Spatial Pyramid Pooling) 模塊來獲取不同 scale 的物體。這使得 DeepLabv3 在處理不同大小的物體時更 robust。

### 3. 性能:

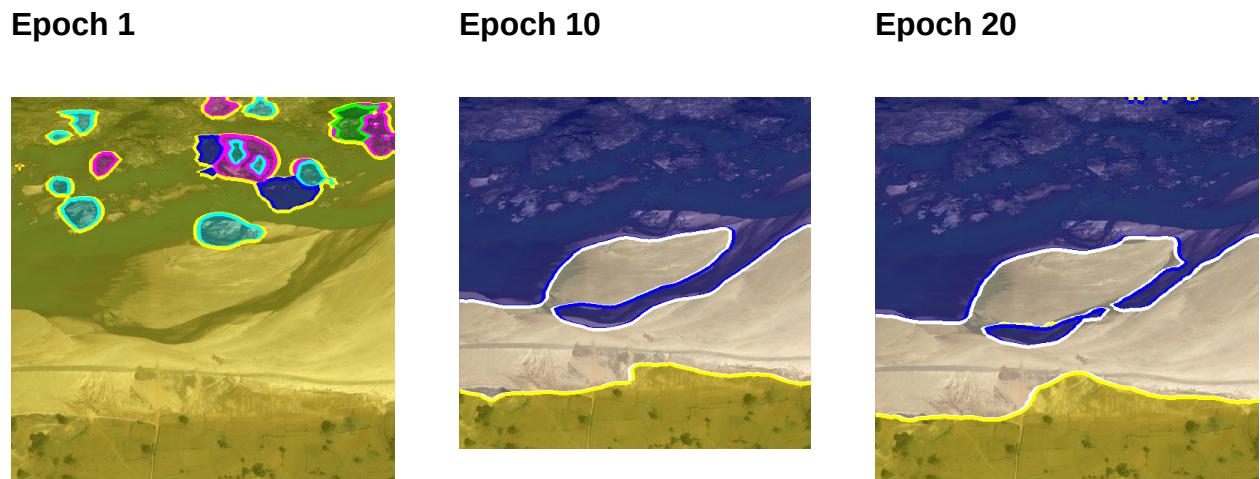
- **VGG16\_FCN32s**: FCN 是早期的語義分割架構之一，並為後續的研究打下了基礎。然而，隨著時間的推移，已經有了更先進的技術 (例如 DeepLab) 來提供更好的性能。
- **DeepLabv3 (ResNet-50)**: DeepLabv3 通常提供了更好的語義分割效果，特別是在邊界區域，這要歸功於其 atrous 卷積和 ASPP module。

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

- A: 0.528
- B: 0.746

### 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.

**0013\_sat.jpg**



**0062\_sat.jpg**



**0104\_sat.jpg**



# Reference

1. ChatGPT 幫忙撰寫程式碼 (Problem 1~3)
2. <https://arxiv.org/pdf/1706.05587.pdf>
3. <https://medium.com/@itberrios6/deeplabv3-c0c8c93d25a4#:~:text=Avaglian%C3%A0%20on%20Unsplash-,Introduction,the%20total%20number%20of%20parameters.>
4. <https://github.com/lucidrains/byol-pytorch>
5. **HUNG-YI LEE (李宏毅) ML Source code**  
[https://colab.research.google.com/drive/15A\\_8iH-6-T3HOmSFrKbjDinBJI-s-16](https://colab.research.google.com/drive/15A_8iH-6-T3HOmSFrKbjDinBJI-s-16)