Final Project IDE20K
CSE318 Deep Vision
邱品諺 B083040029

- 環境建置及平台
  - 使用平台：Google Colab Pro
  - GPU： Tesla P100、Tesla T4

- 使用框架
  - PyTorch 1.11.0

- 資料前處理
  首先需要先了解輸入圖片至丟入模型這之間的過程是如何進行的。

  - 定義分類
    - 將 100 個種類的名稱及 RGB Values 填入 Label List，而本 Project 僅執行種類的分割，因此以 R 值標示各個種類，其餘 G 和 B 皆標示為 0。

  - Transform
    - 將圖片 Resize 至（256, 256），且圖片比對時，僅比對每個 Pixel 的 R 值是否相同。如果相同就將該 Pixel 賦予該種類的 id 值。

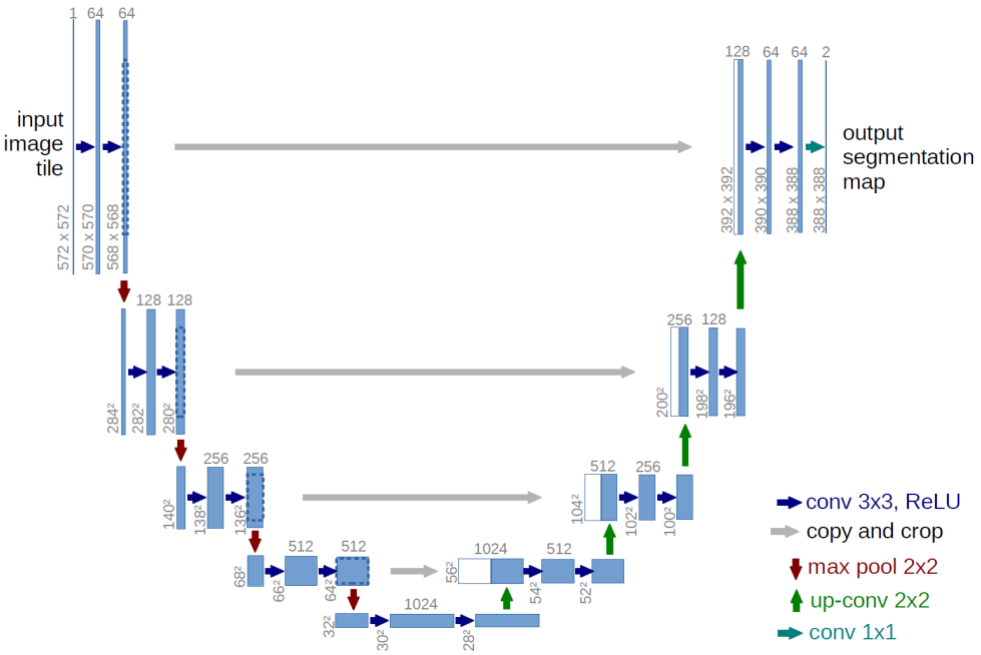
  因此，丟入圖片的模型大小為（256, 256），其中每個 Pixel 標示種類的 id 值，而非 RGB Values。

- 模型建構
  - Model: UNet + Dropout
  - Loss Function: Cross Entropy
  - Optimizer: AdamW
  - Scheduler: ReduceLROnPlateau
  - Weight Initialization: Kaiming Normal

- 訓練方式、過程

- 模型選擇
  - 最初的模型選擇是使用作業 10 的模型架構，利用 mobilenet_v2 作為 Encoder 進行 Feature Extraction，再加上 Upsample 及 Convolution Layer 作為 Decoder 進行 Pixel-wise 分類輸出。發現訓練結果不佳，連 1 張圖片都難以 Overfit，更別說是要訓練 20000 張圖片的 Dataset。因此改變模型架構為 UNet。

  - UNet 為 Encoder、Decoder 的模型架構，且具有 Skip Connection 可有效訓練較深的神經網路。



- 訓練方式

  - 最初訓練的方式非常沒有系統化，因為我直接利用所有的資料丟入模型進行訓練，並隨便挑選了 learning rate 和常見的 optimizer，像是 Adam、SGD + Momentum，再依照訓練結果圖調整 learning rate，如果起初 loss 下降太慢，便調高 learning rate，來來回回調整了很多次浪費了不少時間。

  - 後來回頭審視過去作業訓練的方式，因此從 1 張、10 張等少數張開始 overfit，這個行為看似多餘但其實是檢測模型

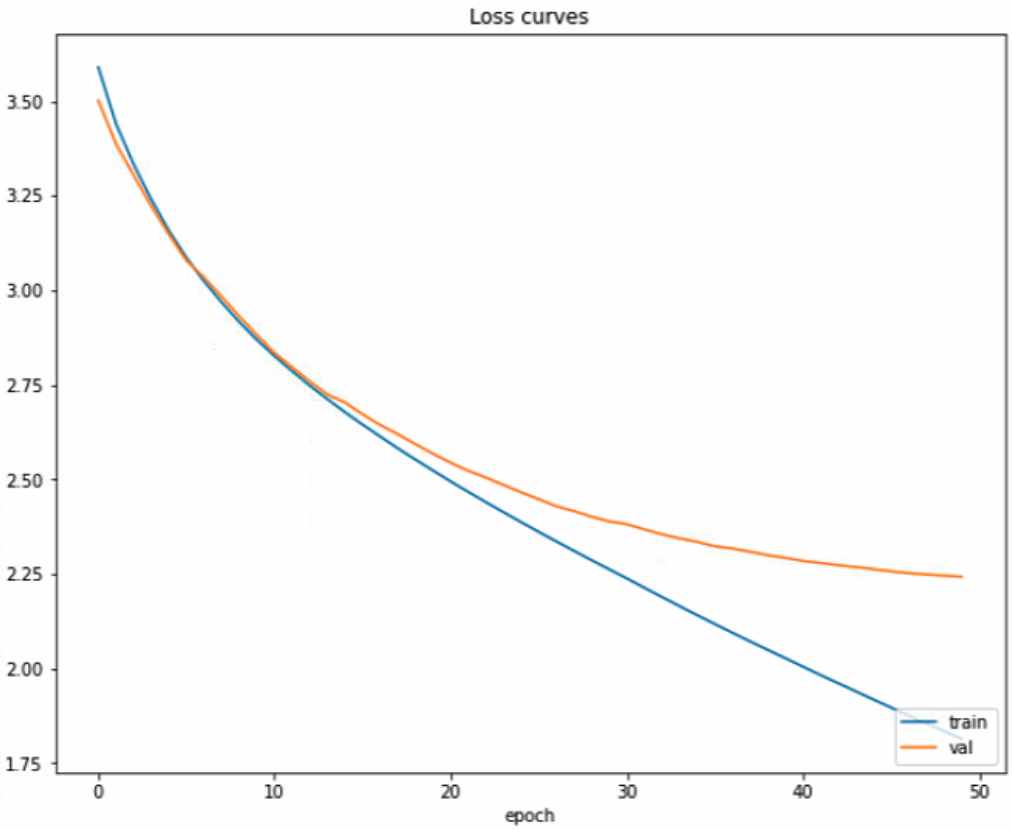
是否能 handle 這個 dataset 的重要方式，並且可自其中學習調整 learning rate 及模型架構的感覺。

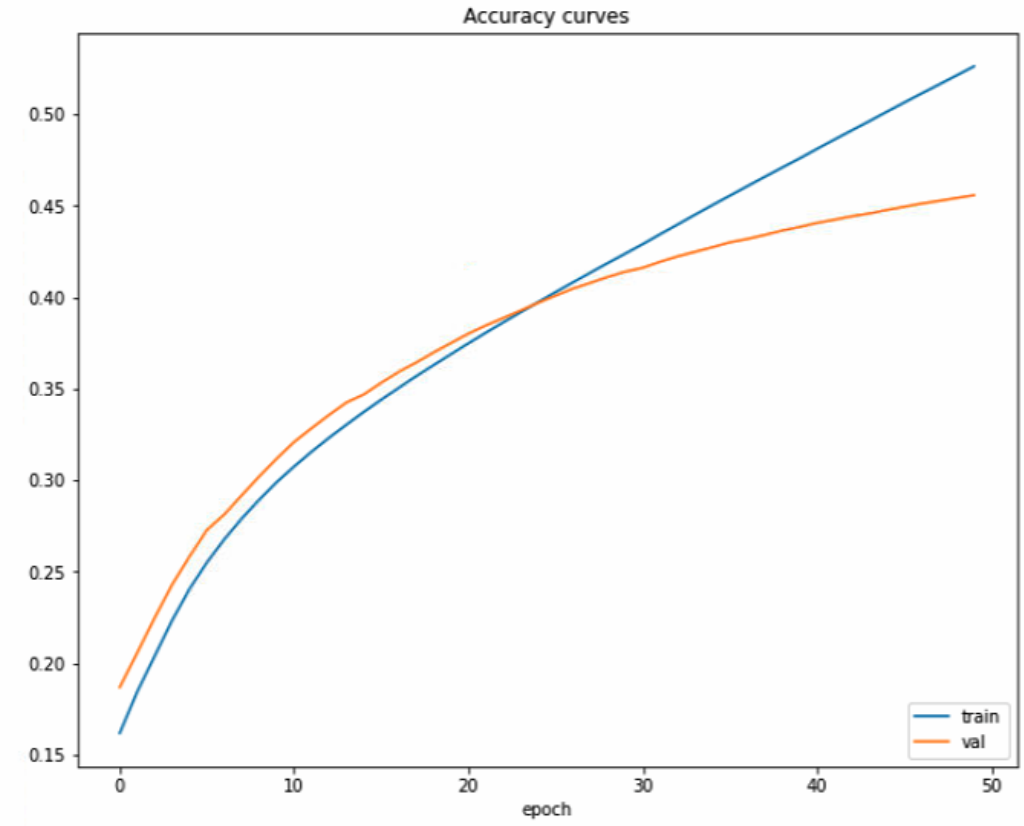
- 最後不免會遇到 Overfitting 的問題，下列為我的處理方式：
  - Dropout
    - 在模型中加了兩層 Dropout 讓模型可以變得較為 General，避免特定 Neuron 學習特定的特徵。
  - Learning rate Scheduler
    - 使用 ReduceLROnPlateau，當 validation loss 沒有下降並達一定次數時就降低 Learning rate。
  - Optimizer
    - 使用 AdamW 取代 Adam，AdamW 大致上可看作使用了 weight decay 的 Adam，可讓模型的訓練權重變得較為平均，使得權重之間的差異較小，以防止 Overfitting 的情況發生。

- 超參數設定
  - Learning rate: 0.002
  - Batch size: 10
  - n_epoch: 60
  - scheduler factor: 0.9
  - scheduler patience: 2

- 訓練結果圖
  - （由於使用 Colab 會有斷線的問題，因此我是將模型儲存並回復 checkpoint 訓練，但由於忘記儲存一些 logger 變數，所以只記錄了到 50 個 epoch 的訓練結果圖，最後 60 個 epoch 訓練完的 Training loss 下降至 1.7 左右，validation loss 下降至 2.23 左右）

  - 下方提供了 50 個 epoch 的 Loss Curve 和 Accuracy Curve，最後訓練至連續五個 epoch 的 validation loss 無法下降時停止。


Loss curves
3.50
3.25
3.00
2.75
2.50
2.25
2.00
1.75
0
10
20
30
40
50
epoch
train
val


Accuracy curves
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
10
20
30
40
50
epoch
train
val

- 訓練驗證平均 IoU
  - Training Mean IoU: 62.867832
  - Validation Mean IoU: 21.602854
  - 下圖為 IoU 計算方式，參考自 [Semantic Segmentation on MIT ADE20K dataset in PyTorch]

```
inters = 0
unions = 0
model.eval()
for i, (img, target) in enumerate(val_data):
    inputs = img.unsqueeze(0)
    inputs = inputs.to(device)

    outputs = model.forward(inputs)
    _, preds = torch.max(outputs, 1)
    pred = preds[0].data.cpu()

    img, target, pred = img.numpy(), target.numpy(), pred.numpy()

    intersection, union = intersectionAndUnion(pred, target, 100)
    inters += intersection
    unions += union

iou = inters / (unions + 1e-10)
print(iou)
print("Mean IoU: %f" %(iou.mean() * 100))
```

- 訓練所遇到的困難
  - Colab 的斷線問題
    - 即使升級 Colab Pro 仍會有斷線的問題，容易會有 runtime disconnected、訓練結果沒儲存的情況發生。
  - 有限的 GPU 資源
    - 利用 Colab 提供的 GPU 訓練 1 個 epoch 可自 40 分鐘到 1 個多小時。
  - 自行訓練權重
    - 不使用 Pretrained model 的權重訓練，自行訓練非常困難。

    - Loss 計算出 NaN
        - 出現 NaN 有可能是 Gradient Explode，或是 numerical stability 的問題，後來發現是因為有可能計算出 1log0 的值導致 NaN，最後利用跳過（continue）的方式解決這個問題。

- 訓練權重
    - https://drive.google.com/file/d/1--4x5s_HFyUssaXkhD4HdVBeWgGTlsP1/view?usp=sharing