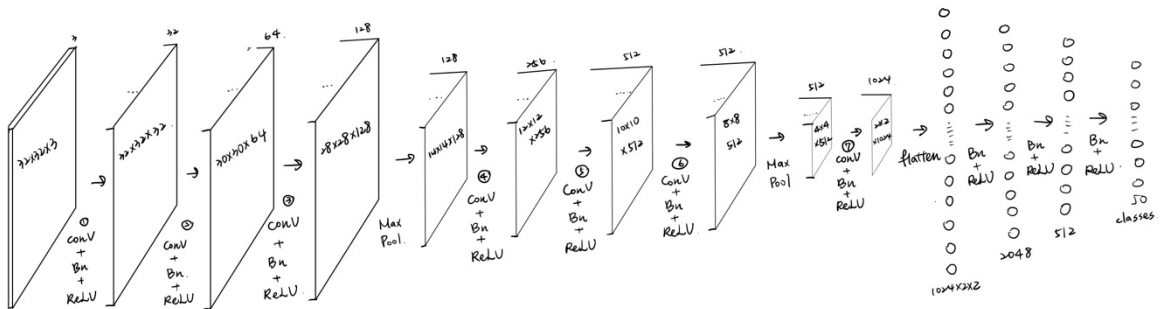


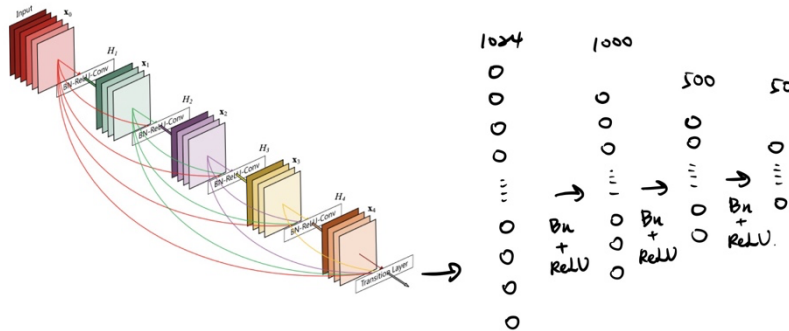
Part 1

1. Draw the network:

A. my CNN from scratch.



B. pretrained weights of DenseNet121



2. Report accuracy on the model:

	Model A	Model B
Accuracy	76%	87%

Model B:

Epoch: 18: 100% 352/352 [05:50<00:00, 1.00it/s, loss=0.605]
 Got 2176/2500, with accuracy 0.87
 Model saved at acc= 0.8703999519348145

3. Report implementation details on model A:

(1) Architecture:

甲、3xConvolution + 1xMaxPool + 3xConvolution + 1xMaxPool + 1x Convolution + fully connected。

乙、因為 image size 僅 32x32，我盡量不使用 padding 以及 pooling layer。為了避免 overfitting，我中間每層皆穿差 Batch Normalization 以及 Dropout(convolution: 0.1, fc: 0.5)。

(2) Criterion and Optimizer:

甲、我使用最基礎的 Cross Entropy Loss 以及 Adam optimizer。

(3) Data augmentation:

甲、我使用了隨機的水平和翻轉、調整亮度、裁切、旋轉、以及使用 ImageNet 的 Data Augmentation 技術。

(4) Hyperparameters:

甲、Batch size = 64,

乙、Learning rate = $1e-4$ for epoch 1 to 320

丙、Learning rate = $1e-6$ for epoch 320 to 400(finetuning)

4. Method B:

甲、因為 Densenet 輸入影像大小為 224x224，因此我先將影像 resize 成 224x224。

乙、Densenet 相較我設計的 CNN 是更深的，有更多的 Convolution 以及 Pooling layers，層與層之間也有許多密集的連階層。

丙、此外，一開始訓練的 weight 是由訓練在 ImageNet 資料及訓練過的，因此應該已經能夠擷取到影像的特徵，實際訓練也是收斂的快非常多。

丁、我在最後加上 3 層 fully connected layers，因為要訓練的資料是 50 個 classes。

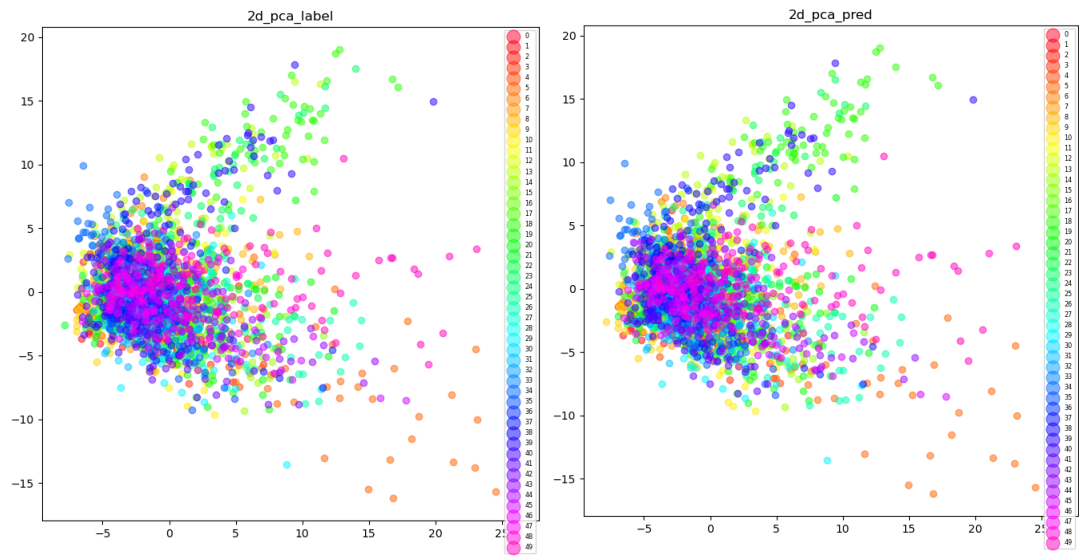
戊、我並沒有將 Densenet 的 weights freeze 起來，而是跟著一起訓練，因為嘗試過這樣子效果較好。

5. Visualize with PCA on Model A on the second last layer

甲、Label：顏色代表正確答案的 label，可看出確實有將正確答案分類成一群一群的，但有些密集分不開，看不出來。

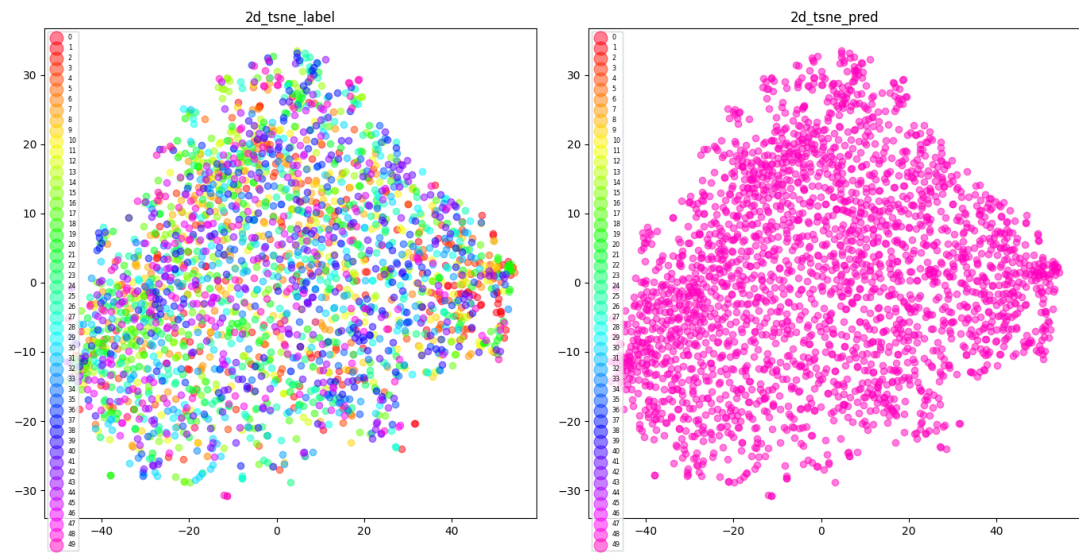
乙、Pred：顏色代表最終 Classifier 預測的結果，可看出相鄰的有被預測成相同類別，和 Label 圖看起來分不差不多。

丙、雖然可大致看出相同顏色的點聚集在一起，並非雜亂的散佈，但全部都聚集成一塊，其實是蠻難區分出每個類別。也許是因為我們只能視覺觀測 2D 平面，而有 50 個 classes，另外顏色也不太夠用。

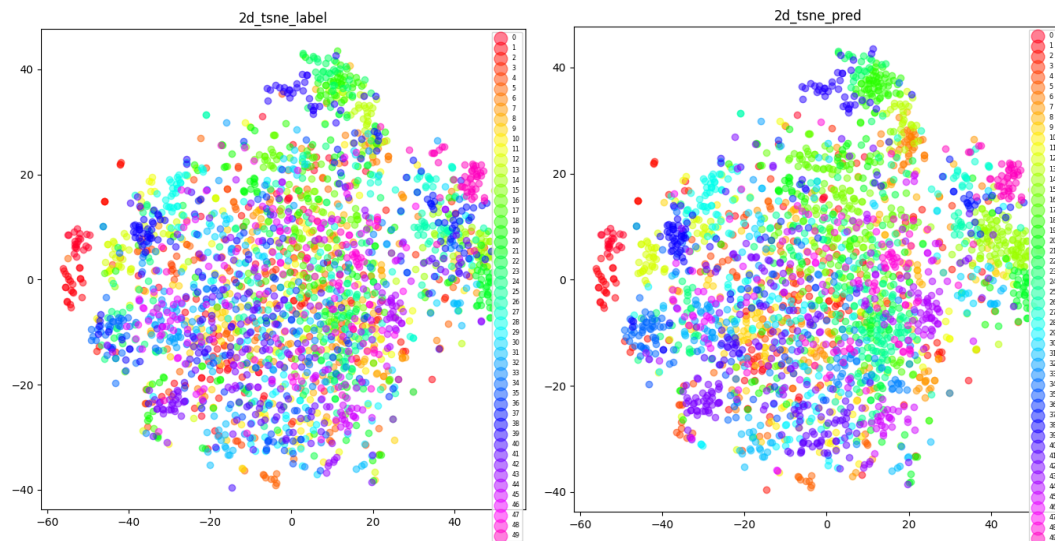


6. Visualize with t-SNE on Model A on the second last layer

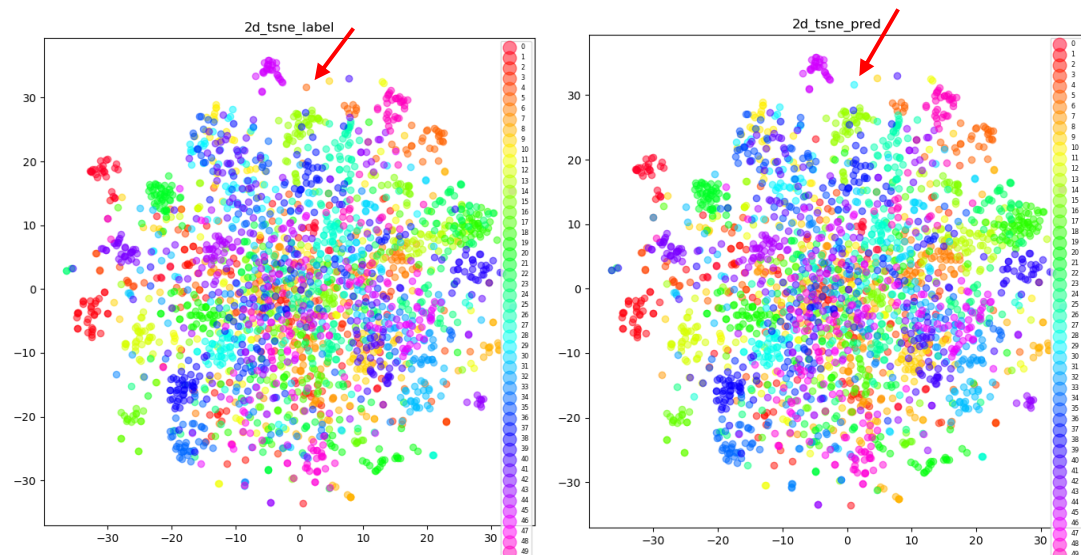
Early (epoch=1):



Middle: (epoch=10, accuracy=45.2%)



Final: (epoch=400, accuracy=76%)



甲、Label 以及 Pred 與上面所述的 PCA 概念相同，相較 PCA 圖更平均的分布，也更能看出顏色的結群情形。

乙、T-SNE 相較 PCA 為非線性的，可看出在 2D 圖上又更能將不同類別的分開，並且有相同類別成一群的現象。

丙、由 3 個 stage 可以看出在一開始時，對於所有 label 皆適區分不出的，散佈平均，因此 predict 的結果無法區分出是哪一個類別。

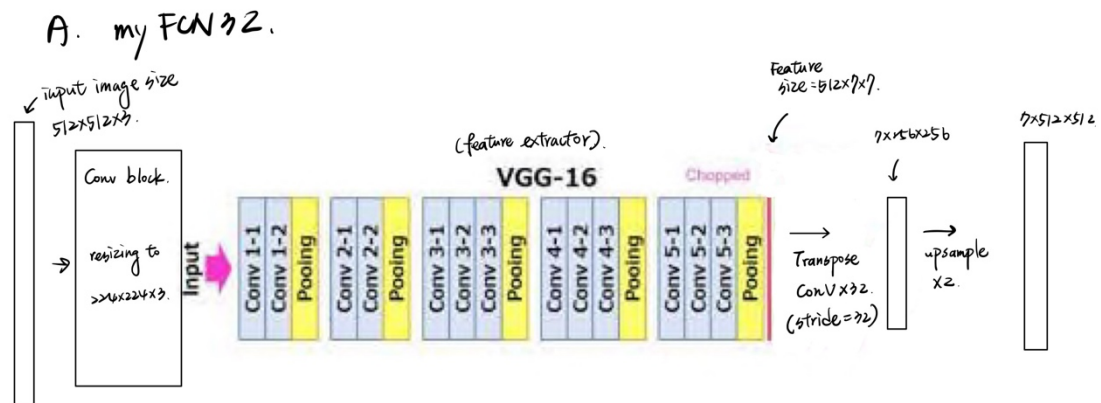
丁、而隨著 epoch 數增加，accuracy 也增加，針對 label 越來越能結成一群一群相同顏色的點，預測也越來越有依據。針對 Middle 以及 Final stage 可以看出 Middle stage 誤判的部分相較多了一些，而 Final stage 的左右兩圖是更相近的。

戊、由箭頭的部分也可以看到的確有誤判的地方，左右並非同一張圖。

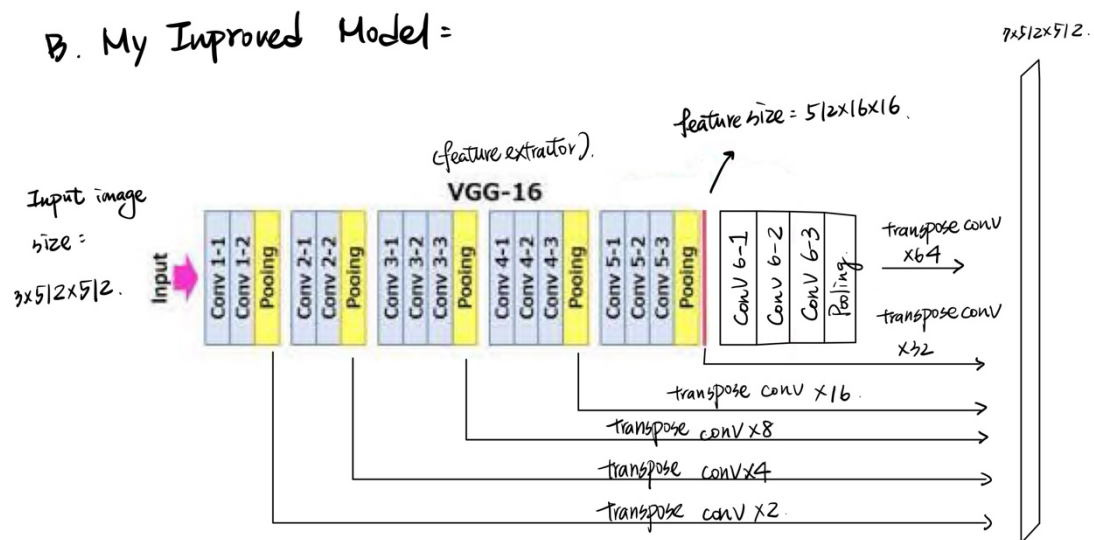
(一共有 50 個類別，而可視的顏色僅有紅色到紫色，已經盡量利用光譜的方式區隔出 50 種顏色，但仍然可能會有顏色相近的類別。)

Part 2

1. Draw the network architecture of my vgg16-fcn32:



2. Draw the network architecture of my improved model:
(inspired by fcn8 and Unet)



3. Report the mean IoU of the two models:

	Vgg16-fcn32	Method B
Mean IoU	60.9	73.3

Method A:


```

Epoch: 44: 100% 32/32 [01:40<00:00, 3.13s/it, loss=0.6]
Epoch: 44: 100% 5/5 [01:21<00:00, 16.32s/it, loss=0.067]
Epoch 44:
class #0 : 0.63646
class #1 : 0.83452
class #2 : 0.18298
class #3 : 0.75876
class #4 : 0.66414
class #5 : 0.58109

mean_iou: 0.609659

Model saved at val meaniou = 0.6096588626733558

```

Method B:

```

Epoch: 24: 100% 250/250 [02:51<00:00, 1.46it/s, loss=0.25]
Epoch: 24: 100% 33/33 [01:22<00:00, 2.49s/it, loss=0.0388]
Epoch 24:
class #0 : 0.76908
class #1 : 0.89555
class #2 : 0.39145
class #3 : 0.81786
class #4 : 0.78807
class #5 : 0.74020


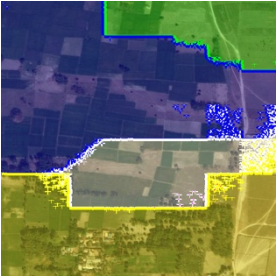
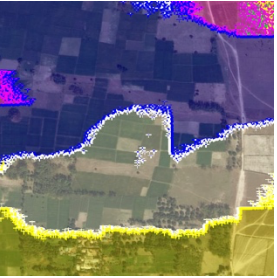

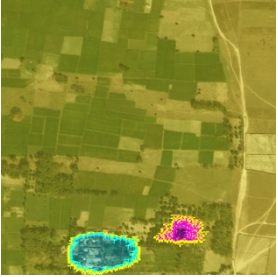
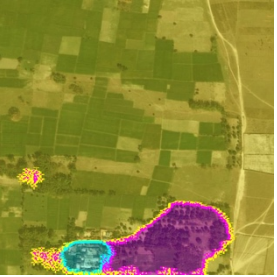
mean_iou: 0.733700



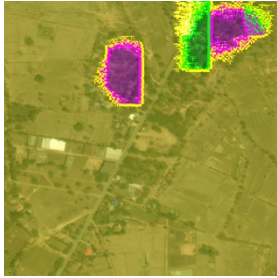
Model saved at val meaniou = 0.7337003749015627|

```

4. Show predicted segmentation masks:

(Method A: fcn32)

	Early(epoch=1)	Middle(epoch=16)	Final(epoch=44)
0013			
0062			

0104			
------	---	--	---