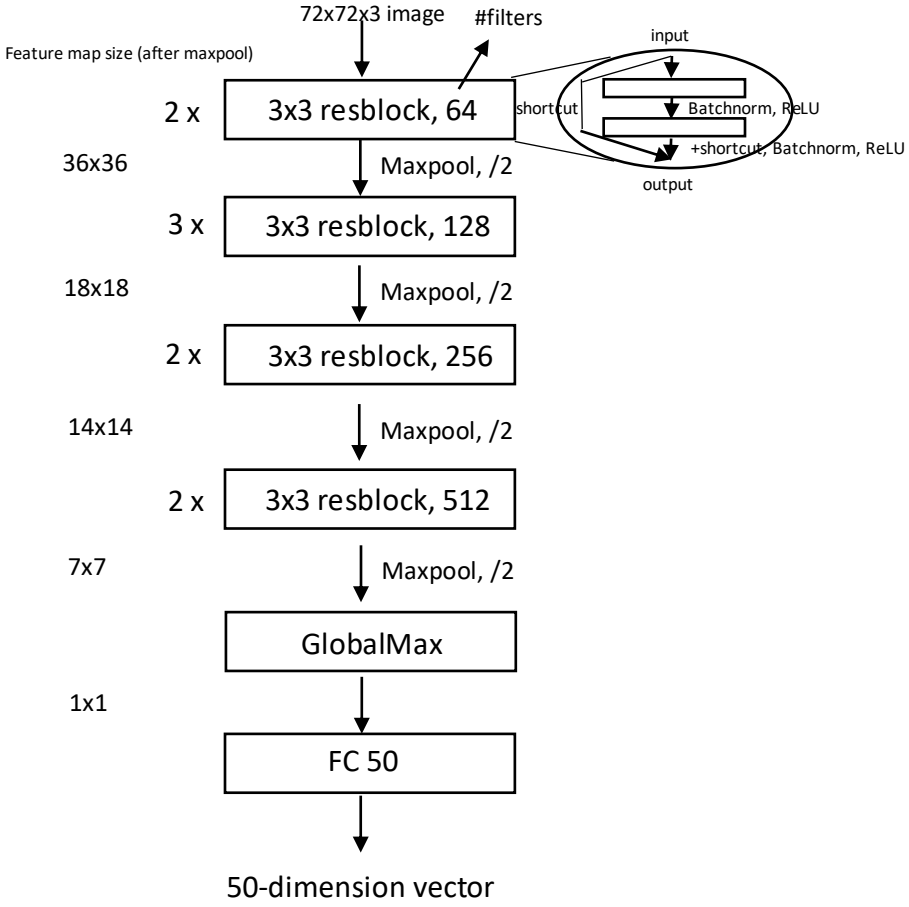
DLCV HW1

工科海洋四 B08505048 劉名凱

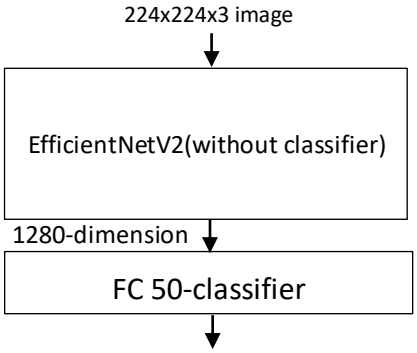
Problem 1

(1)

Model A



Model B



(2)

Model A accuracy: 0.762

Model B accuracy: 0.88

(3)

Preprocessing:

1. Convert pixel value to 0~1
2. Resize the image to (72x72)
3. Normalize the image with (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

Data augmentation:

1. Random horizontal flip p=0.5
2. Random erase part of the image
3. Random rotation from angle (-25 ~ 25 degree)

Optimizer:

RAdam with weight decay=0.01.

Learning rate=0.001 and multiply 0.2 every 40 epochs

Loss function:

Cross entropy which compute the difference between two 50-dimension vector with each entry denote the predicted probability of that class

Cross validation:

Not implemented

I didn't implement any data augmentation at first, the model accuracy stuck at about 0.6. I thought it's because my model was too shallow to tackle this classification problem, so I added few more Resblocks to make network deeper. However, there was no significant progress, and even lower the validation accuracy because the model had overfitted the training data.

After I implemented several data augmentation including RandomCrop() and RandomHorizaontalFlip(), the accuracy started to increase, so I added more augmentation and normalized images. At this point, model's accuracy was about 0.71 but couldn't escalate anymore. Therefore, I tried resizing image to 72x72 and trained with different learning scheduler, and eventually yielded 0.76 accuracy.

(4)

Preprocessing:

1. Convert pixel value to 0~1
2. Resize the image to (224x224)
3. Normalize the image with (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

Data augmentation:

1. Random crop part of the image and resize it to (224x224)
2. Random horizontal flip p=0.5
3. Random erase part of the image
4. Random rotation from angle (-25 ~ 25 degree)

Optimizer:

RAdam with weight decay=0.01.

Learning rate=0.0001 and multiply 0.5 every 2 epoch

Loss function:

Cross entropy which compute the difference between two 50 dimensional vector with each entry denote the predicted probability of that class

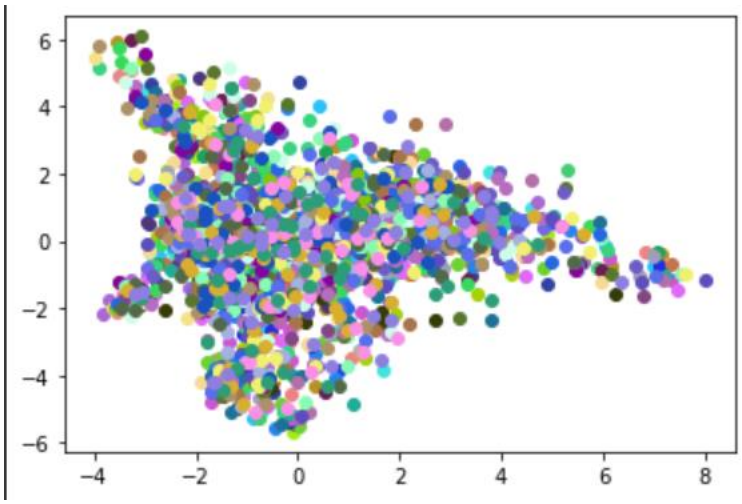
Cross validation:

Not implemented

I used pretrained EFFICIENTNETV2 as backbone to extract feature, then I modified the classifier to output 50-dimension vector. I follow the Pytorch document's instruction regarding how to preprocess images before feed into EFFICIENTNETV2. I freeze all parameters except for the last FC layer and trained with 4 epochs, the accuracy was already 0.78. Then I unfreeze last few layers one by one and trained each with 2 epochs, it only took 10 epoch total to reach final 0.88 accuracy.

The main difference between model A and model B is the building block of the network. I used Resblock as building block in model A, and model B (EfficientNetV2) utilized MBConv and fused MBConv which involve depthwise convolution, squeeze and excitation. Because of depthwise convolution, the computation cost is greatly reduced without sacrificing the accuracy. Therefore, model B's parameters are slightly more than model A's parameters, but achieve much better accuracy (0.76, 0.88)

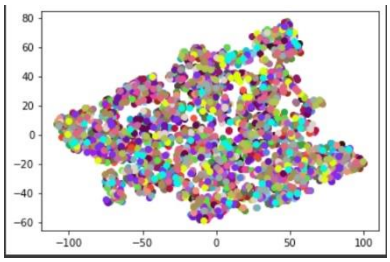
(5)

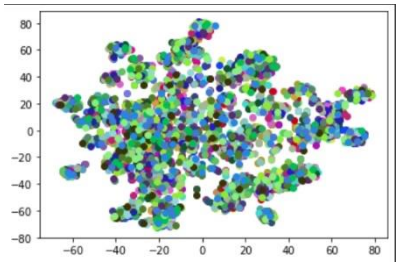


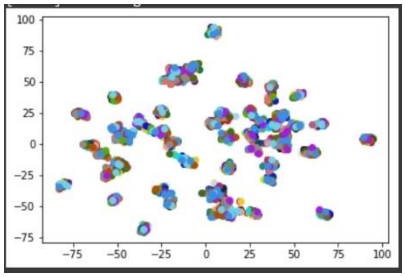
Principal analysis can't effectively distinguish samples with different class, most of the points are mixed together. I believe PCA does not work well with dataset with many classes since PCA method is too simple.

(6) (2500 images)

Epoch: 0 Epoch: 25 Epoch: 80







Unlike PCA, t-SNE effectively allocate samples into different groups and demonstrate a similarity between samples with same class.

At epoch 0, most of the samples are mixed together and the model can hardly differentiate differences between different classes.

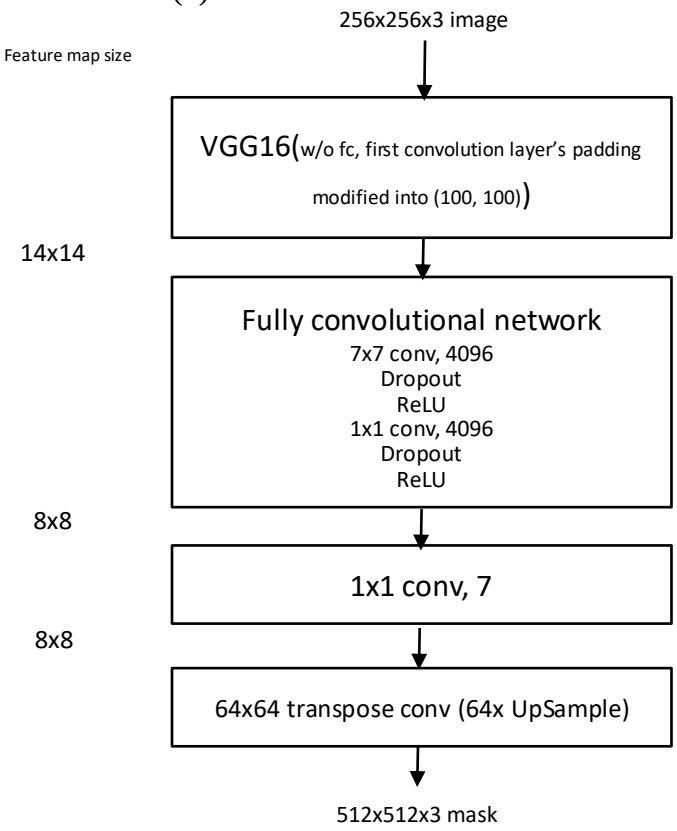
After epoch 25, cluster of points started to show up, accuracy is 0.57.

Epoch 80, accuracy 0.76. Several clusters of points are formed. However, points with same class mixed with other points with different classes. Thus, the model can't fully distinguish the difference between different classes, which can be observed by

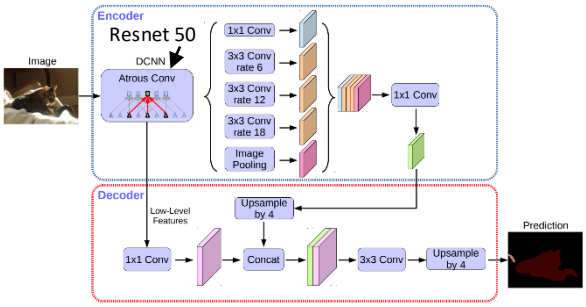
calculating the accuracy.

Problem 2

(1)



(2)



Unlike VGG16+FCN32, Deeplabv3(Resnet-50 as backbone) implements Atrous Spatial Pyramid Pooling which take features with different scale and concatenate them to form a final feature map.

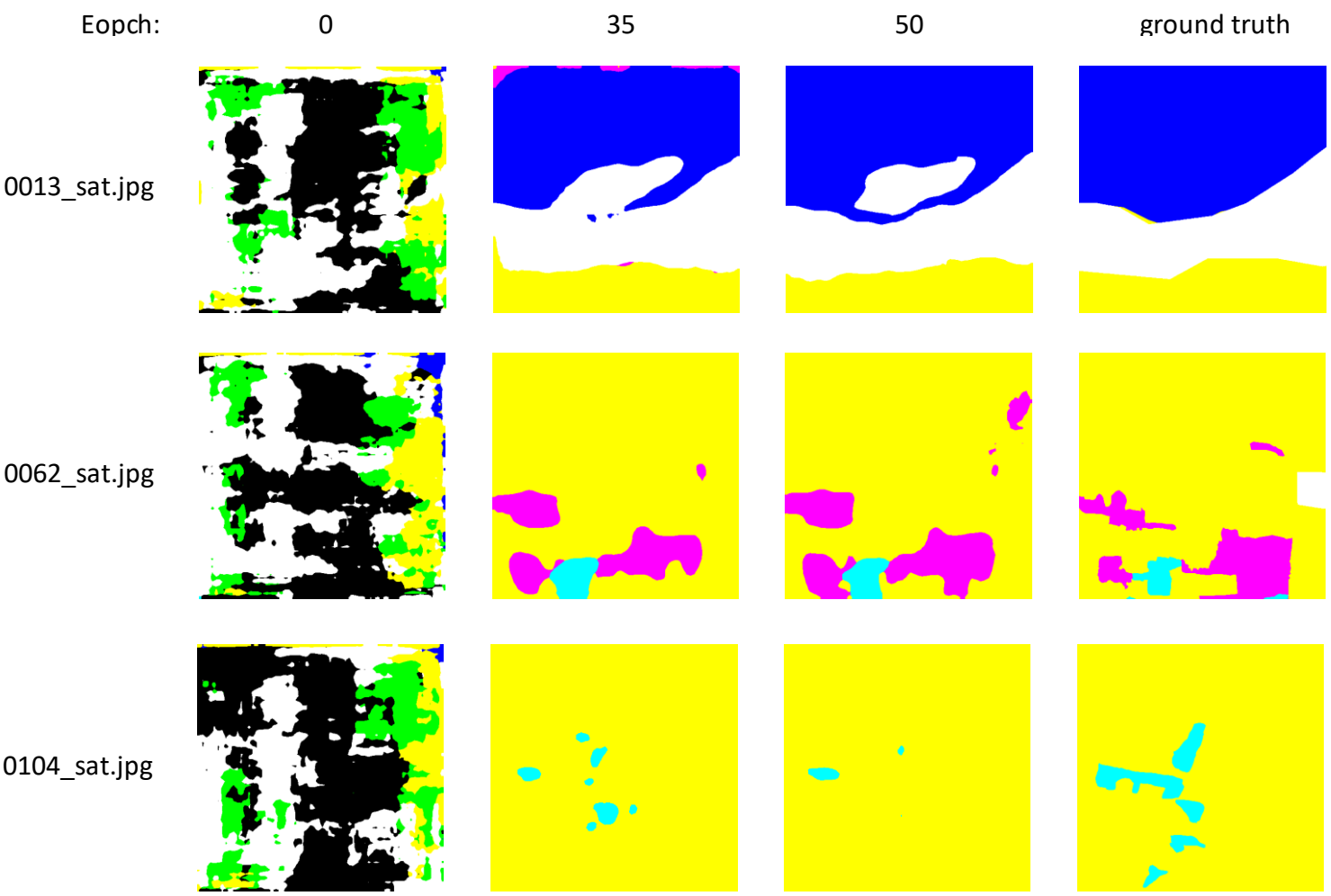
FCN32 only considers the feature map computed by the last layer, thus yield worse result.

(3)

Model A mean IOU: 0.642

Model B mean IOU: 0.74

(4)


Eopch:
0
35
50
ground truth
0013_sat.jpg
0062_sat.jpg
0104_sat.jpg