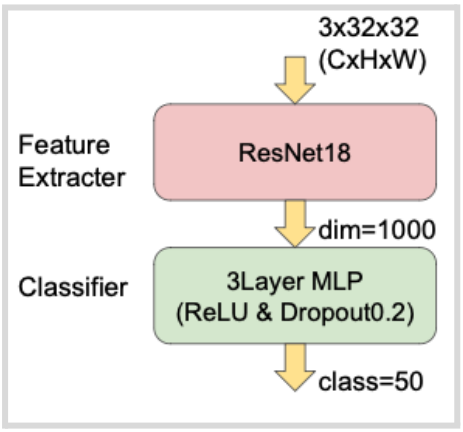
# DLCV HW1

## 1.1

Model A:




## 1.2

Validation Accuracy:

Model A : 64.60%
Model B : 86.28%

## 1.3

- 模型：用了ResNet18作為feature extracter，並將原本的fc層修改成3層的MLP，當中包含兩次ReLU和兩次Dropout(0.2)；
- Loss：CrossEntropyLoss；
- Optimizer：Adam(lr = 0.0001)；
- Batch size: training=32; validation=20
- Scheduler：LambdaLR，當中總共訓練100個epochs，根據以下設定

```
if epoch < warmup_epochs:
    return epoch / warmup_epochs
return 1.0 - (epoch - warmup_epochs) / (total_epochs - warmup_epochs)
```

讓Learning rate乘以這個系數，設定為前30個epochs慢慢上升至1，後70個epochs慢慢收斂

- Data: 對訓練集進行data augmentation－Resize((224,224))RandomHorizontalFlip(p=0.5)、RandomRotation(10)、ToTensor()和Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])，當中resize和normalize是根據torchvision.models所建議
- 儲存方式：儲存validation loss最低的模型而非accuracy的模型

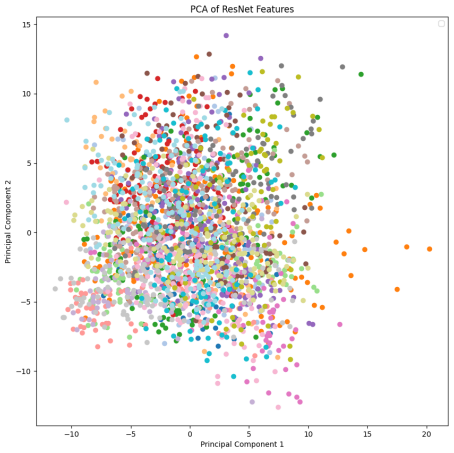
# 1.4

- 所有設定和1.3一樣，只有應用pre-trained weight而已。

# 1.5

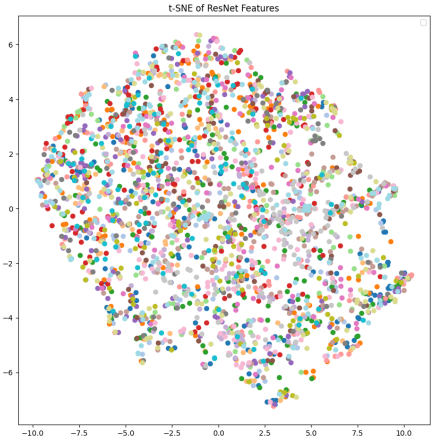
可以看到下圖的PCA visualization的每個Class之間的點都非常接近，與1.6的第100個epoch的t-SNE相比，並沒有明顯看到有分群。
我認為有兩個原因：

1. PCA在分群點視覺化上的能力較t-SNE差
2. 目前此PCA圖所用的模型具泛化能力，而t-SNE第100個epoch的模型已經對training dataset產生overfitting

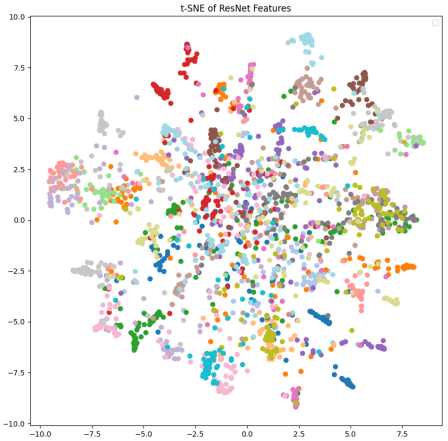



# 1.6

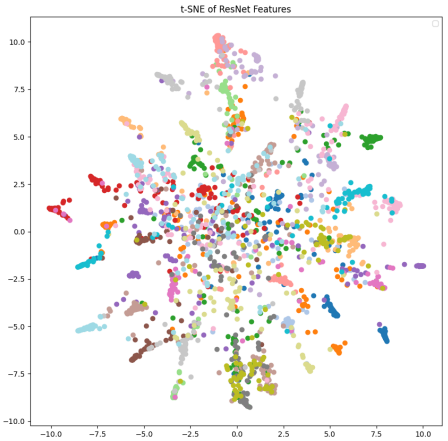
- Epoch 1 (Accuracy: 2.04%, Loss: 3.9176)



- Epoch 50 (Accuracy: 65.00%, Loss: 1.5015)




- Epoch 100 (Accuracy: 70.48%, Loss: 1.7552)




從以上結果可以看到第1個epoch時因為模型未能一次走到loss的最低點，所有數據點都很雜亂的分布在圖上，並沒有任何分群的能力表現；

而第50個epoch可以從相同顏色的點中看到模型已經具備一定的分群能力；

而第100個epoch可以看到大部分點已經能夠清楚分群，但我認為50個epoch和第100個epoch的t-SNE圖差別明顯是因為模型在持續訓練的過程中會對training dataset有overfitting的現像出現，所以accuracy上表現較好，但同時loss也會較高。

# 2.1

- Pretrain setting:直接使用助教推薦的BYOL Github，基本上與原本的BYOL的設置並無不同，只有更改裡面的augmentation function，從

```
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])),
)
```

改成

```
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.Resize(image_size),
    T.CenterCrop(image_size),
    T.Normalize(mean=torch.tensor([0.485, 0.456, 0.406]),std=torch.tensor([0.229, 0.224, 0.225]))
)
```

即只把RandomResizedCrop改成Resize+CenterCrop；

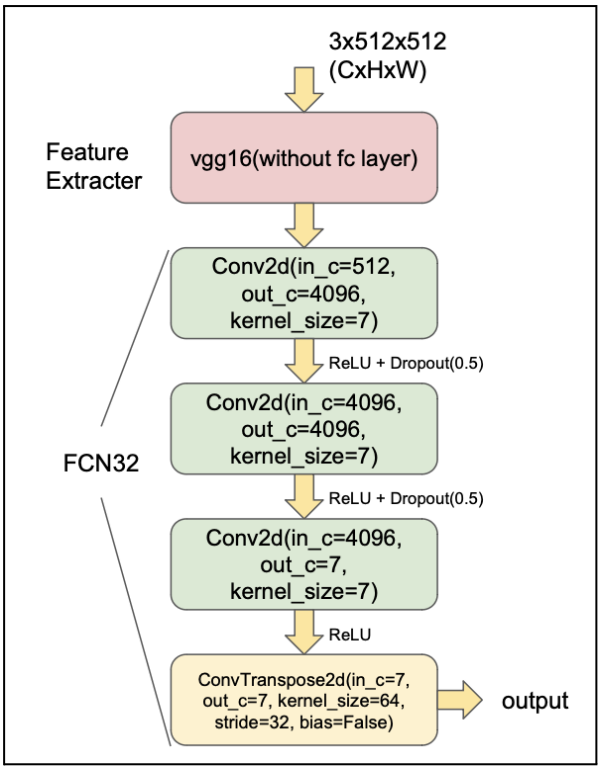
- batch size:100
- optimizer = adam(lr=1e-4)
- 其他都和BYOL的預設值一樣

# 2.2

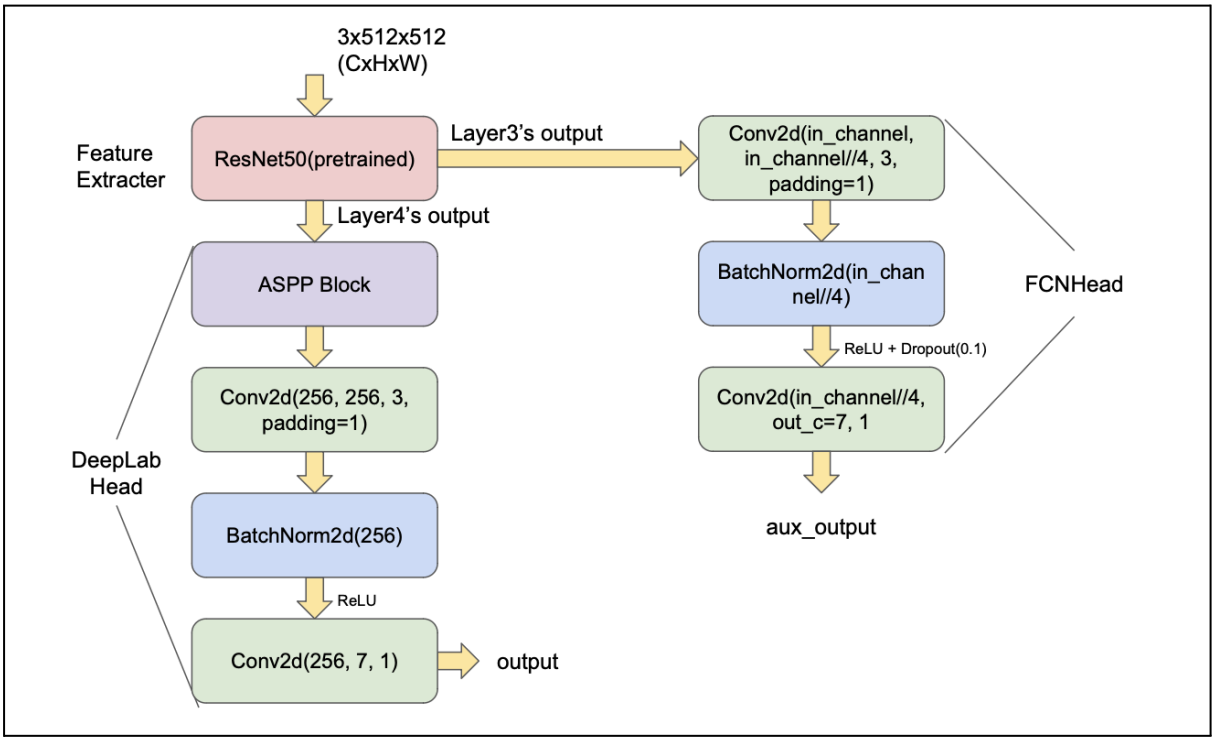
| Setting | Pre-training (Mini-ImageNet) | Fine-tuning (Office-Home dataset) | Validation accuracy (Office-Home dataset) |
|---|---|---|---|
| A | - | Train full model (backbone + classifier) | *32.51%* |
| B | w/ label (TAs have provided this backbone) | Train full model (backbone + classifier) | *45.07%* |
| C | w/o label (Your SSL pre-trained backbone) | Train full model (backbone + classifier) | *43.60%* |
| D | w/ label (TAs have provided this backbone) | Fix the backbone. Train classifier only | *33.25%* |
| E | w/o label (Your SSL pre-trained backbone) | Fix the backbone. Train classifier only | *29.80%* |

# 3.1

- Model architecture

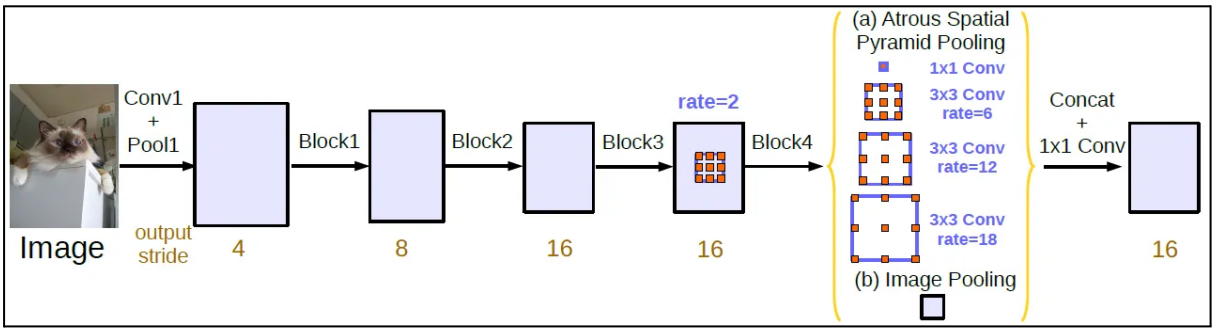


# 3.2




Mode B使用的是DeepLabV3(pretrained)，與Model A主要有3點不一樣

1. Backbone不一樣，ResNet50比vgg16性能更好
2. 使用aux_classifier來增強Backbone extract feature的能力
3. 加入ASPP(如下圖)




# 3.3

mIoUs of model A on the validation set: 0.4525
mIoUs of model A on the validation set: 0.7042

# 3.4

Total epoch=30

依序是0013_sat.png, 0062_sat.png, 0104_sat.png,

- Epoch 1

- Epoch 15

- Epoch30