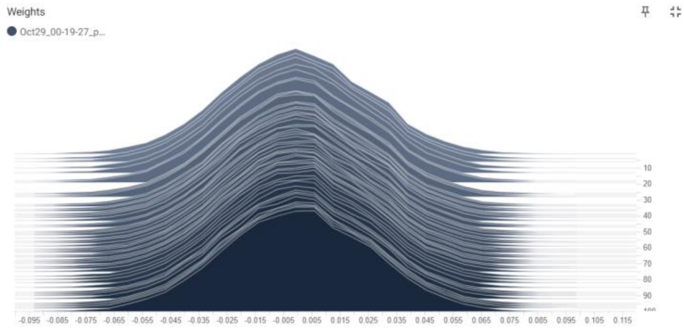
Deep Learning Lab 2 Report

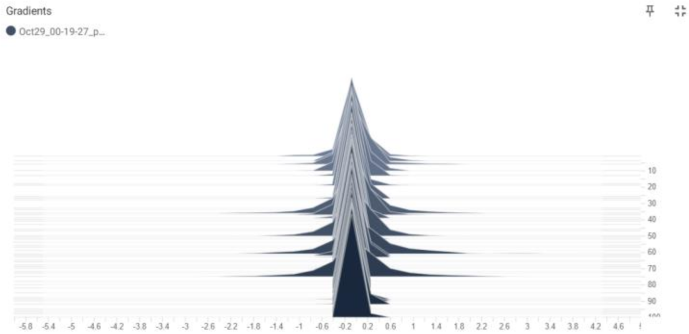
109511207 蔡宗儒

1. Compare resnet18 with and without pretrained

<u>ResNet18 without pretrained</u>

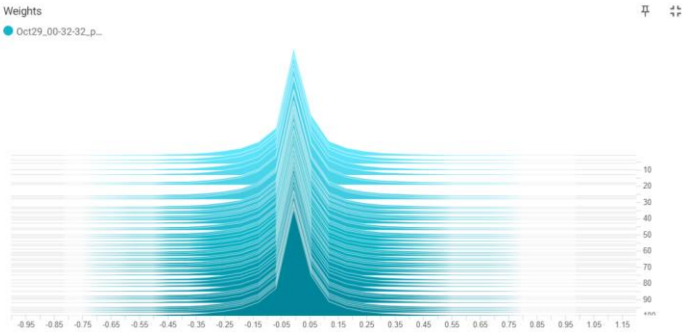
lr = 0.00005,   betas = (0.99,0.999),   weight_decay = 0.00001,   val acc = 86.0606%

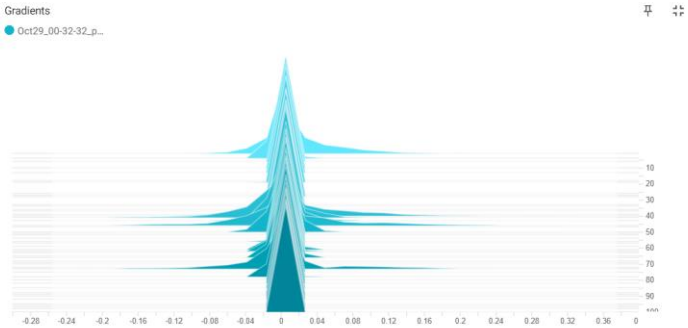




<u>ResNet18 with pretrained weight DEFAULT</u>

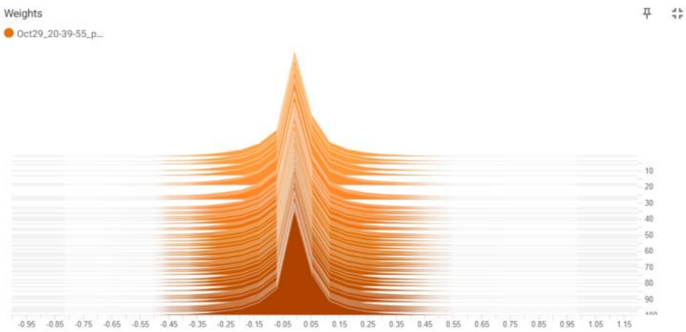
lr = 0.00005,   betas = (0.99,0.999),   weight_decay = 0.00001,   val acc = 98.7879%,   test acc = 89.714%

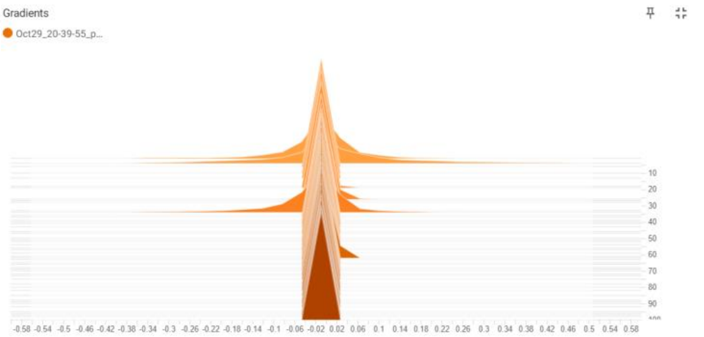




<u>ResNet18 with pretrained weight IMAGENET1K_V1</u>

lr = 0.00005,   betas = (0.99,0.999),   weight_decay = 0.00001,   val acc = 96.9697%,   test acc = 86.857%





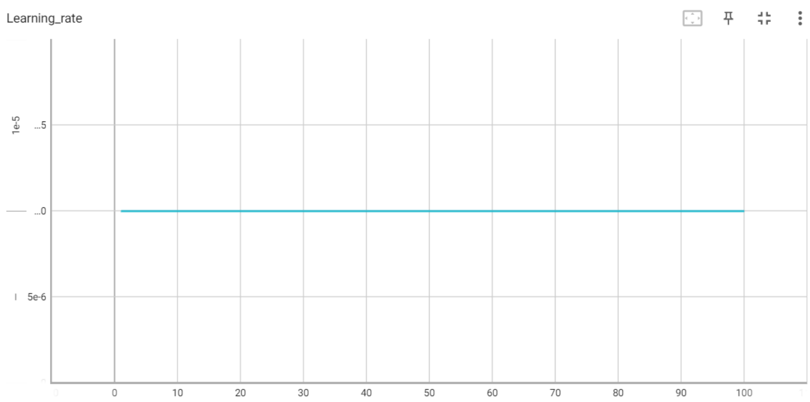
根據結果可發現有 pretrained 過的 model 會有較高的 accuracy，不論是用 DEFAULT 或是 IMAGENET1K_V1，這兩個 pretrained weight 皆能讓 val acc 來到超過 95%以上，而沒有 pretrained 的 model 的 val acc 大約在 85%上下而已，有沒有 pretrained 差了 10%以上。而從 gradient 的 histogram 來看，without pretrained 的 gradient 分布較為分散、變動率較大，即便到了最後幾個 epoch 依然是稍微不均勻的。而 with pretrained 的 gradient 分布較為集中，收斂情況明顯比 without pretrained 更好。這個結果也是可以預想而知的，畢竟 pretrained 的 model 已經先用了別人 train 好的 weight，已經學會一些通用的特徵。這麼一來便可以加快收斂所需的時間，減少訓練時間，且在資料量較少的情況下，優勢會比 without pretrained 更加顯著。

2. Screenshot of task1 (>75% accuracy)

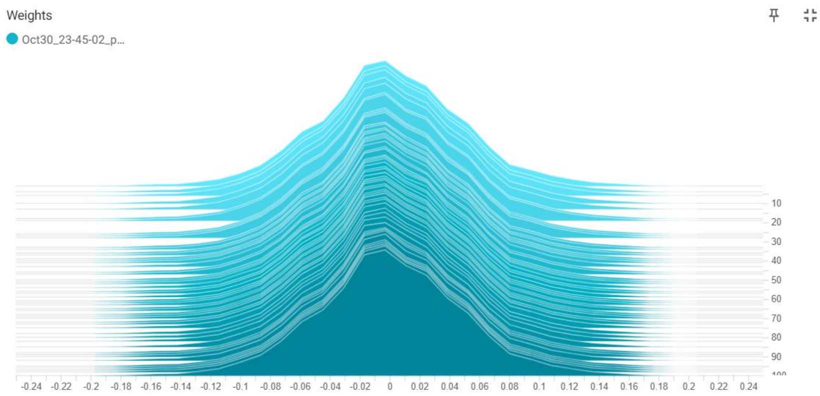
```
epoch: 45
Training Accuracy: 92.3025% Training Loss: 0.2333
Validation Accuracy: 72.1212% Validation Loss: 1.4520
epoch: 46
Training Accuracy: 91.8298% Training Loss: 0.2214
Validation Accuracy: 75.1515% Validation Loss: 1.0105
epoch: 47
Training Accuracy: 93.1803% Training Loss: 0.1954
Validation Accuracy: 79.3939% Validation Loss: 0.6112
epoch: 48
Training Accuracy: 93.7880% Training Loss: 0.1697
Validation Accuracy: 79.3939% Validation Loss: 0.5888
epoch: 49
Training Accuracy: 94.3957% Training Loss: 0.1575
Validation Accuracy: 80.0000% Validation Loss: 0.7070
epoch: 50
Training Accuracy: 94.3957% Training Loss: 0.1546
Validation Accuracy: 80.0000% Validation Loss: 1.2040
```

3. In task2, make graphs for learning rate schedule, weights and gradients (With Tensorboard)

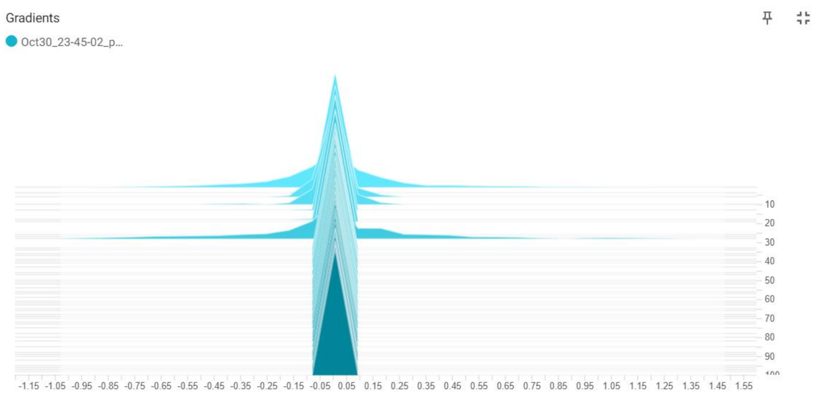
Learning rate schedule



Weights



Gradients



從 gradients 圖可觀察到，前面幾個 epoch，gradients 較為發散，而 training 到後面後逐漸收斂，gradient 分布非常集中。

4. How to improve accuracy

我用 ResNet18、ResNet34、ResNet50、ResNet101 架構，batch size 設為 16，跑 100 個 epoch 得出以下結果。

架構	learning rate	learning rate decay	weight_decay	val acc	test acc
resnet18 with pretrained weight DEFAULT	0.00001	None	0.00001	96.3636%	89.714%
resnet18 with pretrained weight DEFAULT	0.00005	None	0.00001	98.7879%	89.714%
resnet18 with pretrained weight DEFAULT	0.00005	Exponential rate decay	0.00001	98.1818%	86.571%
resnet18 with pretrained weight IMAGENET1K_V1	0.00005	None	0.00001	96.9697%	87.428%
resnet18 with pretrained weight IMAGENET1K_V1	0.00005	Exponential rate decay	0.00001	98.7879%	86.857%
resnet34 with pretrained weight DEFAULT	0.00001	None	0.00001	97.5758%	88.000%
resnet34 with pretrained weight DEFAULT	0.00005	None	0.00001	97.5758%	90.285%
resnet34 with pretrained weight DEFAULT	0.00005	Exponential rate decay	0.00001	97.5758%	90.857%
resnet34 with pretrained weight IMAGENET1K_V1	0.00005	None	0.00001	98.7879%	89.428%
resnet34 with pretrained weight IMAGENET1K_V1	0.00005	Exponential rate decay	0.00001	96.9697%	84.571%
resnet50 with pretrained weight DEFAULT	0.00001	None	0.00001	98.1818%	88.571%
resnet50 with pretrained weight DEFAULT	0.00005	None	0.00001	97.5758%	91.142%
resnet50 with pretrained weight DEFAULT	0.00005	Exponential rate decay	0.00001	99.3939%	90.000%
resnet50 with pretrained weight IMAGENET1K_V2	0.00005	None	0.00001	98.7879%	91.142%
resnet50 with pretrained weight IMAGENET1K_V2	0.00005	Exponential rate decay	0.00001	98.7879%	92.000%
resnet101 with pretrained weight DEFAULT	0.00005	None	0.00001	98.7879%	90.000%
resnet101 with pretrained weight DEFAULT	0.00005	Exponential rate decay	0.00001	98.7879%	91.428%
resnet101 with pretrained weight IMAGENET1K_V2	0.00005	None	0.00001	96.9697%	86.571%
resnet101 with pretrained weight IMAGENET1K_V2	0.00005	Exponential rate decay	0.00001	98.7879%	91.714%

可以發現這之中 resnet50 with pretrained weight IMAGENET1K_V2 有最好的效果，但可能是我超參數設置的不夠好，才讓 resnet101 沒有跑出它應有的效果。從中也可以觀察到，越複雜的架構普遍會有更高的 accuracy，而複雜的架構做 learning rate decay 看起來成效會比用在簡單一點的架構更有用。我最後採用的 model 是 swin_v2_s，lr 設為 0.00001，沒有做 learning rate decay，跑出來的 test acc 為 94.857%，而我發現用這個 model 每次跑出來的結果都會不一樣，這是因為此架構有包含多層 Dropout Layer，如下圖。Dropout 引入了隨機性和機率的概念，才會導致每次結果都不太一樣。

```
model
```

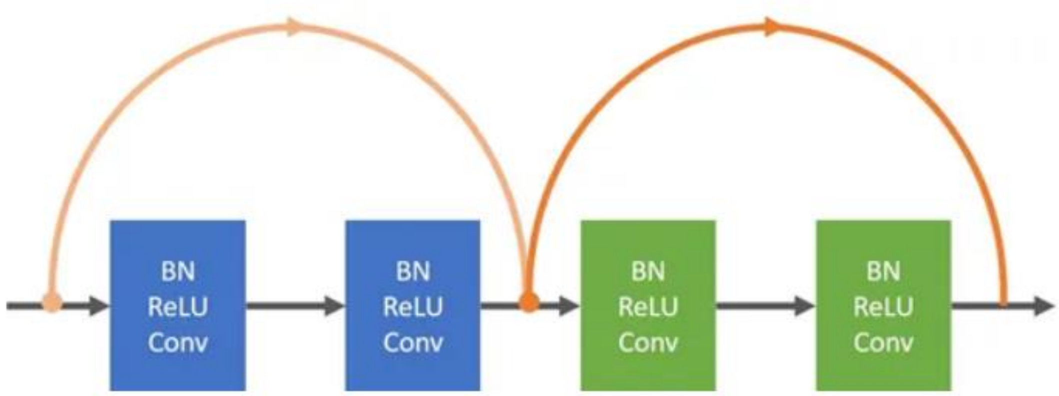
```
        (qkv): Linear(in_features=96, out_features=288, bias=True)
        (proj): Linear(in_features=96, out_features=96, bias=True)
        (cpb_mlp): Sequential(
          (0): Linear(in_features=2, out_features=512, bias=True)
          (1): ReLU(inplace=True)
          (2): Linear(in_features=512, out_features=3, bias=False)
        )
      )
      (stochastic_depth): StochasticDepth(p=0.013043478260869565, mode=row)
      (norm2): LayerNorm((96,), eps=1e-05, elementwise_affine=True)
      (mlp): MLP(
        (0): Linear(in_features=96, out_features=384, bias=True)
        (1): GELU(approximate='none')
        (2): Dropout(p=0.0, inplace=False)
        (3): Linear(in_features=384, out_features=96, bias=True)
        (4): Dropout(p=0.0, inplace=False)
      )
```

另外可以將 if accuracy > acc_best 改為 if accuracy >= acc_best，如下圖。這樣當出現相同 val acc 時，可以儲存更充分訓練後的 model，有可能可以提高準確率。

```
if accuracy >= acc_best:
    acc_best = accuracy
    print("model saved")
    torch.save(model, "model.pth")
```

5. ResNet 架構

當網路層深度增加後，越容易發生梯度消失的問題，所以 ResNet 就在 2015 年被提出來了。ResNet 印入了 Residual Block，能夠跳躍連接，讓訊息可以直接在網路中跳躍傳遞，將輸入特徵和輸出特徵相加， 這麼一來即便某一層發生梯度消失也不會連帶影響到其他層的 weight。



6. Reference

https://reurl.cc/Xm3eV0
https://reurl.cc/OjDXE7