學號：r07921001　　系級：電機所二　　姓名：李尚倫

1. 請從 Network Pruning/Quantization/Knowledge Distillation/Low Rank Approximation 選擇兩個方法(並詳述)，將同一個大 model 壓縮至同等數量級，並討論其 accuracy 的變化。 (2%)

這題中用到的大 Model 的結構如下圖，是一個自己接的簡單的 CNN model (以下稱 Big model)：6 layer CNN + 1 layer FC model train from scratch 100 ep using Adam

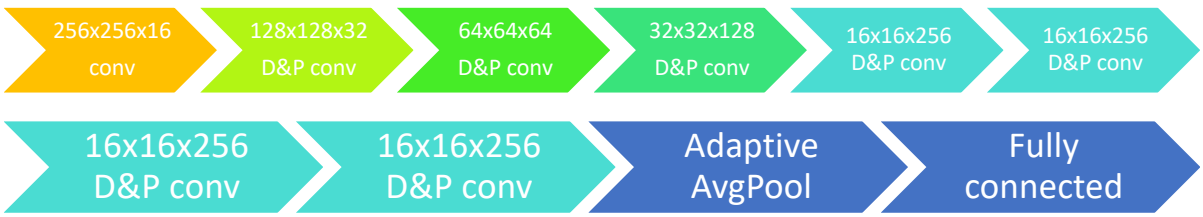
```
TeacherNet(
  (cnn): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU()
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (12): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU()
    (15): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (17): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (18): ReLU()
    (19): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (20): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (21): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (22): ReLU()
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc): Sequential(
    (0): Linear(in_features=4096, out_features=11, bias=True)
  )
)
```

而我們使用以下兩種方法來壓縮此 Model 至同一個數量級，結果如下表：

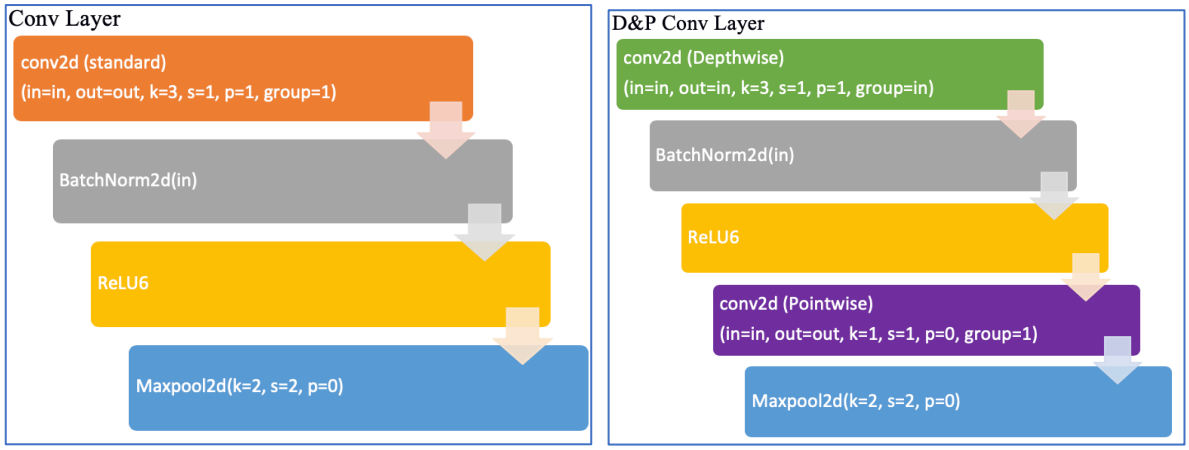
| Method | Validation Accuracy | File size (B) | Model's parameters |
|---|---|---|---|
| Big model | 0.7988 | 6,477,626 | 1,615,627 |
| Knowledge Distillation with Low Rank Approximation | 0.7901 | 1,047,430 | 256,779 |
| Directly Quantization | 0.7913 | 1,623,152 | 1,615,627 |

Knowledge Distillation with Low Rank Approximation 的方法是使用 Low Rank Approximation 的方式去設計一個輕量的 Student model 用 Knowledge Distillation 的方式來學 Teacher model (即為 Big model)，Low rank 的 Student model 的設計如下：

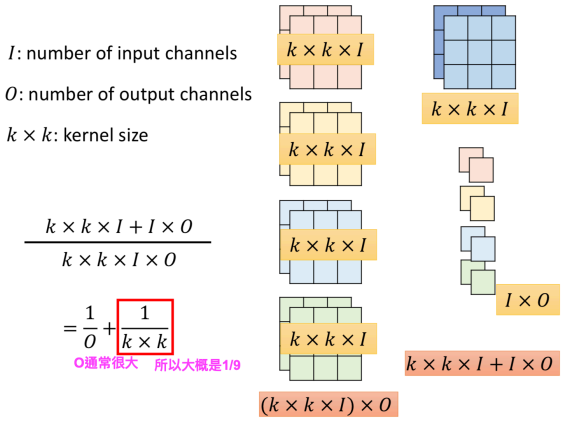
Student_model：(inputs=256x256x3, output=11)



一般的 Convolution Layer weight 數量很大，僅在第一層使用(通常第一層都需要是一般的 conv)，而 D&P 的 DW(Depthwise Convolution Layer)則是各 channel 的 feature map 各過自己的 filter 處理後，再用 PW(Pointwise Convolution Layer)把所有 channel 的 feature map 的單個 pixel 資訊合在一起。如下圖是 Student_model 的單個 layer 展開：




如此 D&P 的做法可以大量減少參數，做出一個輕量化的 model，圖示如下圖，且 accuracy 比單純只是參數少的 CNN 來的更高(詳見第四題討論)：




而輕量化的 student model 通常會搭配 teacher model 一起訓練會比 train from scratch 的效果好(詳見第二題討論)，因此搭配 Knowledge Distillation 的方法，讓 student model 在訓練時不只和原本 data 的 label 算 loss 也和 teacher model 的 output 算 loss：

$$\text{Loss} = \alpha T^2 \times \text{KLDivLoss}\left(\frac{\text{Teacher's Logits}}{T} \,||\, \frac{\text{Student's Logits}}{\text{T}}\right)$$
$$+(1-\alpha)\text{CrossEntropy}(\text{Student's Logits}, \text{Data's Label})$$

上式為設計的 Loss，用 Adam optimizer lr=0.001 並 training 100ep 後的結果，參照上面的 table 顯示，結果和 Big model 接近，無論檔案大小或參數上的壓縮效果都不錯。

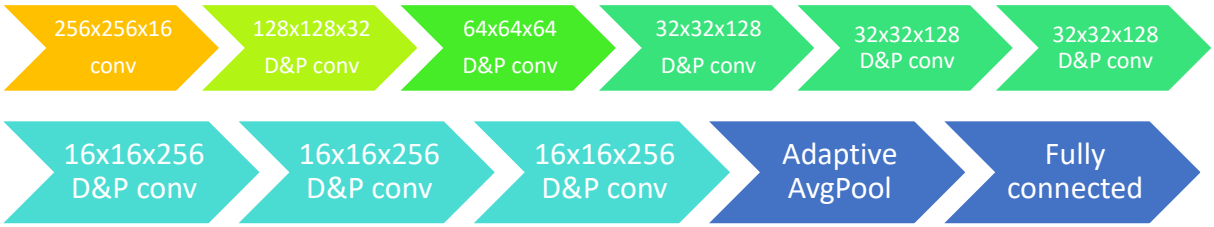
而 Directly Quantization 則是直接壓縮 Big model 的大小，把原本的訓練好的 Float32 的參數，32bit 改降存成 8bit，使用的是 numpy.unit8，轉換的方式是根據上下界做 normalize 後 map 到 0~255 的區間，公式如下：

$$W' = round\left(\frac{W - \min(W)}{\max(W) - \min(W)} \times (2^8 - 1)\right)$$

儲存上下界與轉換完的 unit8 weight，並存成.pkl 檔，檔案的大小可以縮小四倍左右，並保有不錯的 accuracy，藉此可以觀察出 weight 的 resolution 其實並沒有很高，只需要 0~255 搭配上下界就可以很趨近了，但相較於前者的方式，這個方式僅降低檔案大小，並沒有減少模型實際的參數量，因此計算速度不會增快。

由上面的比較可知，兩者各有優缺點，所以最後 kaggle 上的結果我們使用兩者混合的方式，可以既減少參數增加 predict 的速度，也可以大量減少檔案大小。先用 Low Rank Approximation 的方式重新設計了一個 Student model 比原本助教提供的更多層，並調整 model 結構，然後再用 Knowledge Distillation 以助教提供的 ResNet18 ImageNet pretrained & fine-tune 當作 Teacher net 來訓練，以 0.002 的 learning rate 搭配 Adam optimizer，訓練 200 個 epoch，得到還不錯的 accuracy，然後再下調 learning rate 到 0.001 訓練 3 個 epoch，accuracy 仍有上升，繼續下調 learning rate 到 0.0005，訓練 2 個 epoch，就到極限了(後續還有再降低 learning rate 或訓練多一點 epoch 並沒有獲得更好的效果)，得到第一段的 compressed model (.bin model)。此階段的 model 仍有 918038 Bytes 還不到門檻，再使用 Quantization 把參數從 32bit 的資料型態用前面提到的方式採存成 8bit，一口氣把大小降到 236937 Bytes，為第二階段 compressed model (.pkl model)，在 Kaggle 的 accuracy 有 0.8494 超過 strong baseline，並有在條件的大小內。

Student_model_deeper：(inputs=256x256x3, output=11)




實驗的 Table 如下，最後 submit 為 student_model_deeper_203ep：

| Model / epoch | .bin<br>acc<br>on val | .pkl<br>acc<br>on val | .pkl<br>acc<br>on kaggle | model<br>parameters | .pkl<br>file size (B) |
|---|---|---|---|---|---|
| student_custom_small<br>(助教提供的小 model) | 0.8137 | 0.8087 | 0.8344 | 256779 | 268471 |
| student_model/202ep | 0.8207 | 0.8155 | 0.8332 | 256779 | 268471 |
| student_model_deeper/200ep | 0.8283 | 0.8277 | 0.8446 | 224011 | 236937 |
| **student_model_deeper/203ep** | **0.8388** | **0.8362** | **0.8494** | **224011** | **236937** |
| student_model_deeper/205ep | 0.8429 | 0.8376 | 0.8470 | 224011 | 236937 |

以下三題只需要選擇兩者即可，分數取最高的兩個。

2. [Knowledge Distillation] 請嘗試比較以下 validation accuracy (兩個 Teacher Net 由助教提供) 以及 student 的總參數量以及架構，並嘗試解釋為甚麼有這樣的結果。你 Student Net 的參數量必須要小於 Teacher Net 的參數量。(2%)

x. Teacher net architecture and # of parameters:
torchvision's ResNet18, with 11,182,155 parameters.
y. Student net architecture and # of parameters:
使用第一題的 Student_model, with 256,779 parameters.

a. Teacher net (ResNet18) from scratch: 80.12%
b. Teacher net (ResNet18) ImageNet pretrained & fine-tune: 88.51%
c. Your student net from scratch: 79.94%
d. Your student net KD from (a.): 79.80%
e. Your student net KD from (b.): 82.13%

| Role (model) training method | Validation accuracy | Model size (B) | Parameters | Filename |
|---|---|---|---|---|
| a. Teacher net (ResNet18) from scratch | 0.8012 | 44,788,712 | 11,182,155 | teacher_resnet18_from_scratch.bin |
| b. Teacher net (ResNet18) ImageNet pretrained & fine-tune | 0.8851 | 44,788,712 | 11,182,155 | teacher_resnet18.bin |
| student_custom_small | 0.8137 | 1047430 | 256779 | student_custom_small.bin |
| c. Student net (Student_model) from scratch | 0.7994 | 1047574 | 256779 | student_model_from_scratch.bin |
| d. Student net (Student_model) KD from a | 0.7980 | 1047574 | 256779 | student_model_from_scrateacher.bin |
| **e. Student net (Student_model) KD from b** | **0.8213** | **1047574** | **256779** | **student_model_200ep.bin** |

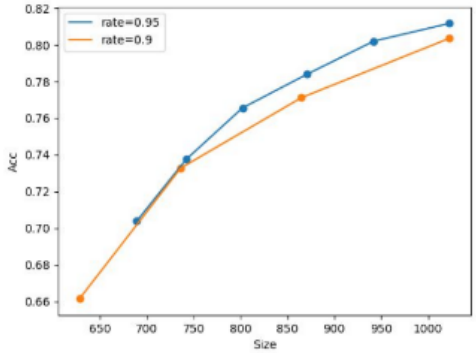
使用第一題描述的 KD 方式分別以兩種不同的 teacher model weight 的去訓練第一題的 student model，並和 train from scratch 做比較，其中都 train 200 個 epoch，都使用 Adam optimizer with lr=0.001。由結果可以看出，很明顯的雖然對 student model train from scratch 的效果還可以，但還是有 teacher model 做 KD 的準確性更高，而且還必須是好的 teacher model 效果才會比較好。原因就如同老師上課所講的，第一，當 data 不

是很乾淨的時候，對一般的 model 來說他是個 noise，只會干擾學習。透過去學習其他大 model 預測的 logits 會比較好，能朝已知對的方向收斂。第二，可以給予 label 和 label 之間的關連，teacher model 有每個分類的 score，而不是像 ground truth 的 one hot encoding，這可以引導小 model 去學習，讓學習的效果更好。例如在數字辨識數字 8 可能就和 6,9,0 有關係。因此有 teacher model 會比 train from scratch 好，而 teacher model 本身的分類效果(accuracy)也必須要好，才能夠正確引導 student model 往好的方向收斂，所以 KD from b 會比 KD from a 好。

3. [Network Pruning] 請使用兩種以上的 pruning rate 畫出 X 軸為參數量，Y 軸為 validation accuracy 的折線圖。你的圖上應該會有兩條以上的折線。(2%)

(Skip) (refer to hw slide)

如果我們用兩種rate去prune (每次都prune掉5%和10%), 其參數量和Accuracy的圖可能就會長的像下圖。這題就是請你們畫出這張圖。




被 prune 掉的 neuron 越多，雖然 model 的大小變小了運算速度也會比較快，但 accuracy 也會有顯著的下降，因為被 prune 過的 model 並不能完全代表原本的 model，且刪掉 neuron 的依據是按重要性，刪越多(rate 越大)越有可能刪到重要的，因此當一直迭代下去越剪枝越多(rate*rate*rate...)，pruned model 和原 model 的失真就越大。

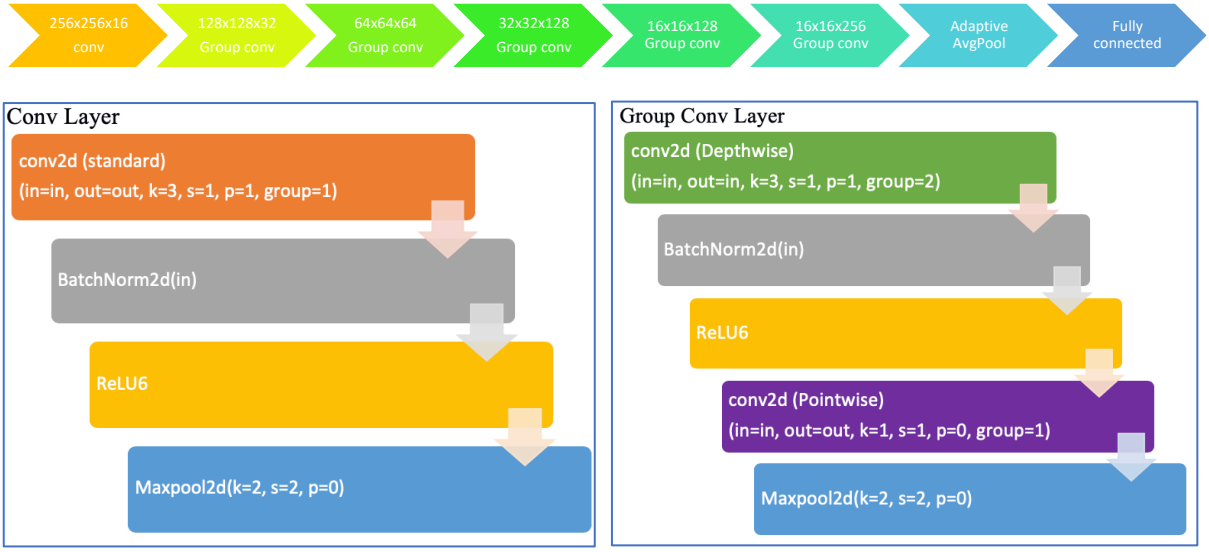
4. [Low Rank Approx / Model Architecture] 請嘗試比較以下 validation accuracy，並且模型大小須接近 1 MB。 (2%)

a. 原始 CNN model (用一般的 Convolution Layer)

```
FullCnnNet(
  (cnn): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU()
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (12): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU()
    (15): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (16): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (17): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (18): ReLU()
    (19): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (20): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (21): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (22): ReLU()
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc): Sequential(
    (0): Linear(in_features=2048, out_features=11, bias=True)
  )
)
```

b. 將 CNN model 的 Convolution Layer 換成參數量接近的 Depthwise & Pointwise
(參見第一題的 Student model)

c. 將 CNN model 的 Convolution Layer 換成參數量接近的 Group Convolution Layer
(Group 數量自訂，但不要設為 1 或 in_filters)




而下方 table 則是上面三個 model train from scratch 100ep with Adam lr=0.001 的結果

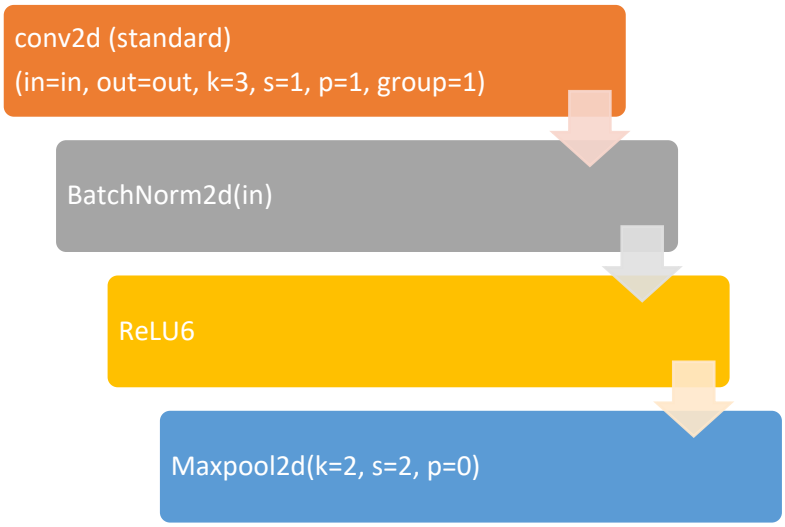
| | Validation acc | Model size (B) | Paramters | Model structure |
|---|---|---|---|---|
| Standard Convolution Layer | 0.7615 | 1120786 | 277611 | 見 4.a. |
| Depthwise & Pointwise Convolution Layer | 0.7834 | 1047430 | 224011 | 見 4.b. |
| Group Convolution Layer | 0.7429 | 958468 | 236571 | 見 4.c. |

一般的 convolution layer 都是在講 group=1 的時候，即一個 filter 的大小就是 kernal_size* kernal_size*input channel 數，參數量多。而 group=input channel 的時候，一個 filter 的大小是 kernal_size* kernal_size*1，如果搭配 output channel 數 = input channel 數的話，即是第一題提到的 Depthwise 的 convolution ，搭配 Pointwise 使用的話，要達到相同效果的 convolution 相較一般法參數量可以減少很多。group=2 的意思則介於兩者之間，一個 filter 的大小是 kernal_size* kernal_size* input channel 數/2，參數量介於兩者之間，和原本的一般法之間比較沒有直接的等效關係。

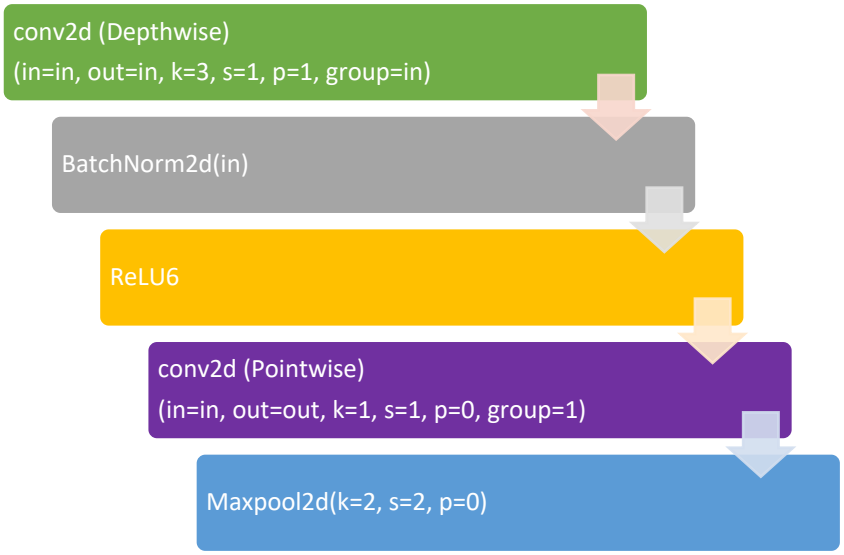
而上邊表格紀錄了，所設計的三種不樣的 convolution model 的 performance，其中 D&P 的方法表現得最好，因為在一樣參數量(model size)的情況下，他所等效(代表)的 model 實際上是要用更多參數量才能在一般 convolution model 上實作的，而至於 group convolution model 的表現原本預期是應該介於一般與 D&P 之間的，估計是因為 model 的接法上並沒有考慮太多的設計，沒有像 D&P 的方法在 Depthwise 後搭配一個剛好的 Pointwise，做出一個可以合理解釋的 model，所以 performance 沒有特別好。

圖源：

Conv Layer




D&P Conv Layer




Group Conv Layer