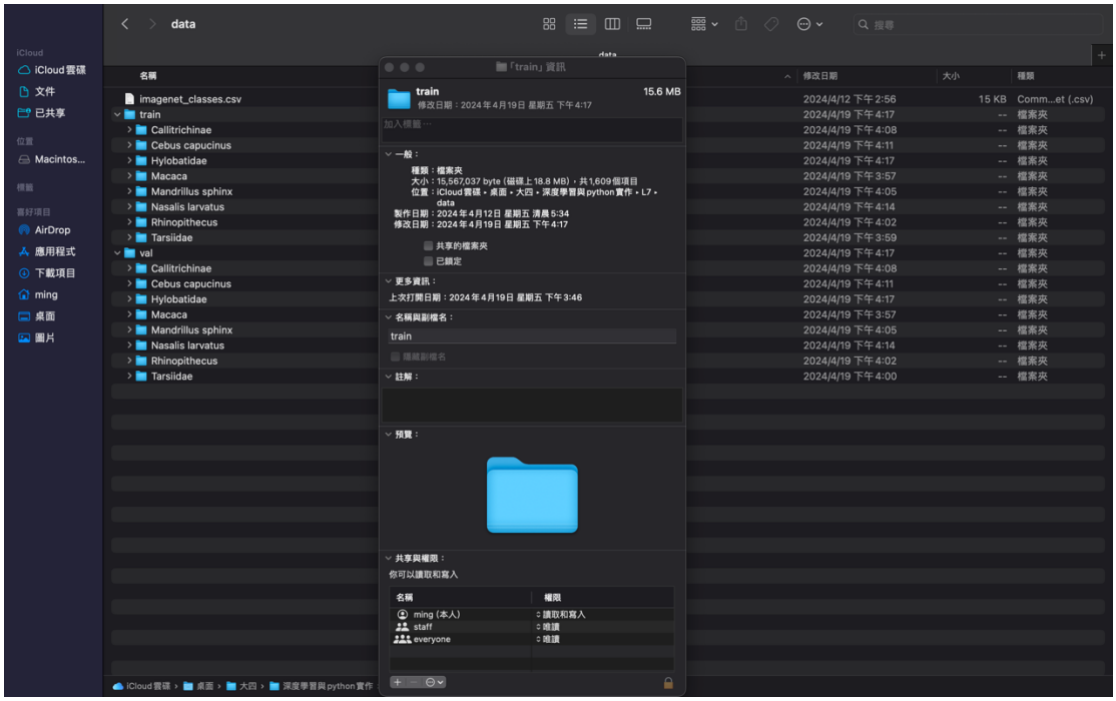
Exercise:

Continuing from the previous assignment, collect at least 200 images for each category:

1. Design a program to compare the prediction results.
2. Select one model and use different weights (at least 4 weights) to compare the prediction results.
3. Choose 6 models (one weight for each model) and introduce your chosen models with graphics/tables and text, comparing the prediction results.

PS. At least 8 pages of A4 paper, font size 12, Arial font, line spacing 1.5.

Data distribution



10 categories, 200 images for each category

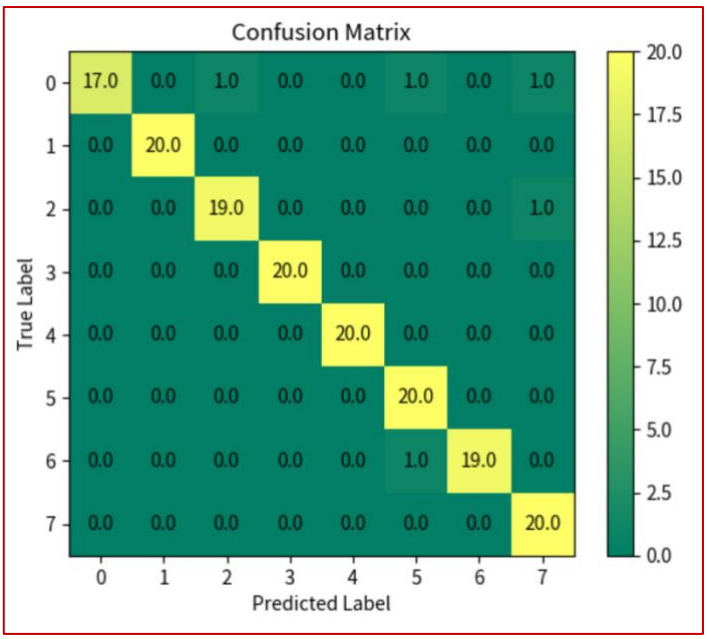
1 model and 4 different weights:

Model: ResNet

4 different weights:

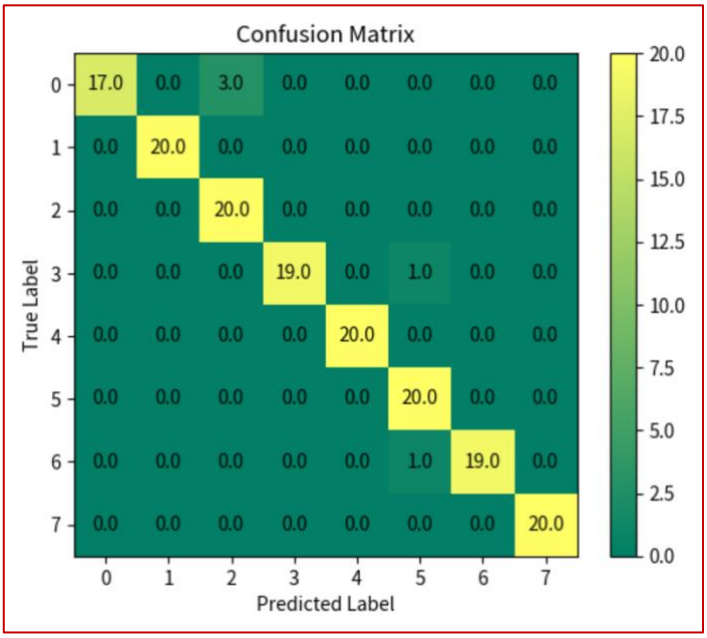
1. resnet18

```
epoch: 0, Train loss: 0.2221, acc:0.7194,     Val loss: 0.0387, acc:0.9563
epoch: 1, Train loss: 0.1481, acc:0.8131,     Val loss: 0.0277, acc:0.9625
epoch: 2, Train loss: 0.1237, acc:0.8419,     Val loss: 0.0281, acc:0.9563
epoch: 3, Train loss: 0.0977, acc:0.8813,     Val loss: 0.0214, acc:0.9750
epoch: 4, Train loss: 0.0873, acc:0.8831,     Val loss: 0.0251, acc:0.9688
```



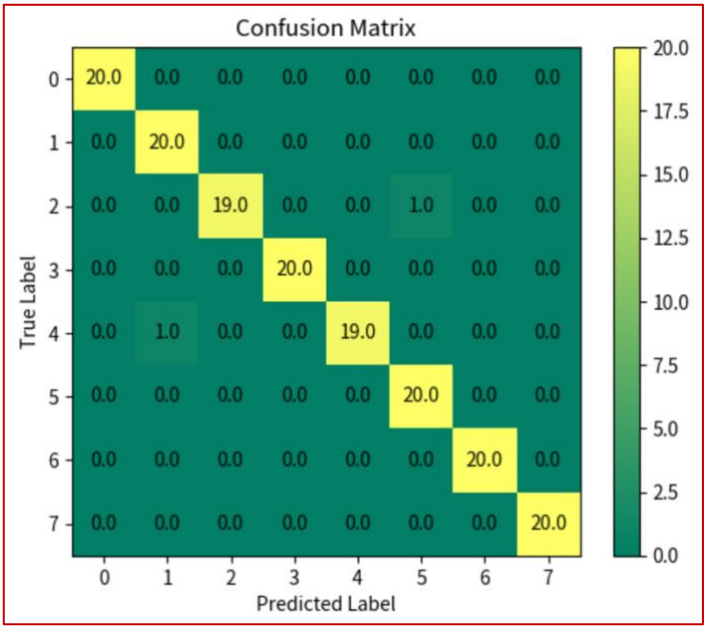
2. resnet34

```
epoch: 0, Train loss: 0.2288, acc:0.7081,     Val loss: 0.0327, acc:0.9625
epoch: 1, Train loss: 0.1344, acc:0.8381,     Val loss: 0.0212, acc:0.9688
epoch: 2, Train loss: 0.1041, acc:0.8775,     Val loss: 0.0188, acc:0.9750
epoch: 3, Train loss: 0.0851, acc:0.9019,     Val loss: 0.0266, acc:0.9688
epoch: 4, Train loss: 0.0851, acc:0.8963,     Val loss: 0.0331, acc:0.9688
```



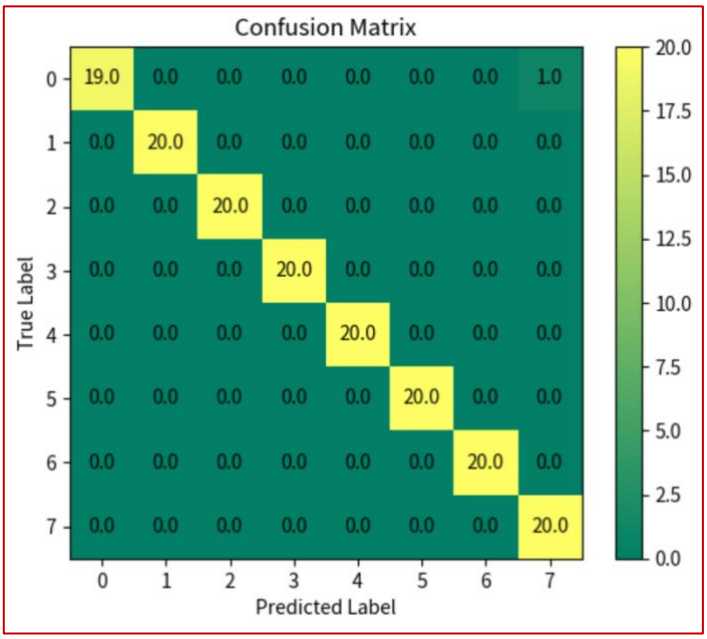
3. resnet50

```
epoch: 0, Train loss: 0.2755, acc:0.6819,     Val loss: 0.0742, acc:0.9437
epoch: 1, Train loss: 0.1381, acc:0.8306,     Val loss: 0.0278, acc:0.9812
epoch: 2, Train loss: 0.1030, acc:0.8719,     Val loss: 0.0199, acc:0.9875
epoch: 3, Train loss: 0.0807, acc:0.9019,     Val loss: 0.0141, acc:0.9875
epoch: 4, Train loss: 0.0753, acc:0.9094,     Val loss: 0.0120, acc:0.9875
```



4. resnet101

```
epoch: 0, Train loss: 0.2337, acc:0.7225,     Val loss: 0.1273, acc:0.8625
epoch: 1, Train loss: 0.1081, acc:0.8650,     Val loss: 0.0318, acc:0.9688
epoch: 2, Train loss: 0.0852, acc:0.8975,     Val loss: 0.0173, acc:0.9812
epoch: 3, Train loss: 0.0646, acc:0.9194,     Val loss: 0.0123, acc:0.9875
epoch: 4, Train loss: 0.0535, acc:0.9331,     Val loss: 0.0118, acc:0.9938
```



6 models

1. ResNet101

Introduction:

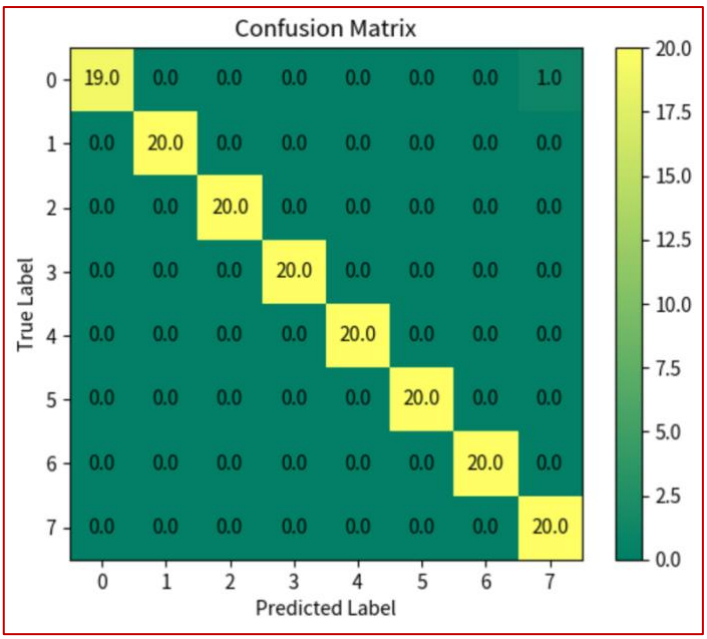
ResNet101 的主要創新之處在於引入了殘差連接（residual connections），這種連接方式可以加速信息在網絡中的傳播，從而解決了傳統深度神經網絡中的梯度消失（vanishing gradient）和梯度爆炸（exploding gradient）等問題。殘差連接通過將輸入直接添加到網絡的後續層輸出中，允許模型學習殘差（即輸入與目標之間的差異），而不是直接學習目標值，這樣可以更容易地訓練非常深的網絡。

Structure:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Result:

```
epoch: 0, Train loss: 0.2337, acc:0.7225,     Val loss: 0.1273, acc:0.8625
epoch: 1, Train loss: 0.1081, acc:0.8650,     Val loss: 0.0318, acc:0.9688
epoch: 2, Train loss: 0.0852, acc:0.8975,     Val loss: 0.0173, acc:0.9812
epoch: 3, Train loss: 0.0646, acc:0.9194,     Val loss: 0.0123, acc:0.9875
epoch: 4, Train loss: 0.0535, acc:0.9331,     Val loss: 0.0118, acc:0.9938
```



2. GoogLeNet

Introduction:

主要創新之一是引入了稱為 Inception 模塊的結構，這是一種高效的多尺度特徵提取方法。Inception 模塊使用了多個不同大小的卷積核，並且在同一層內進行了併發運算，以捕獲不同尺度下的圖像特徵。這樣的結構使得模型具有更好的特徵提取能力，同時還能保持計算效率。

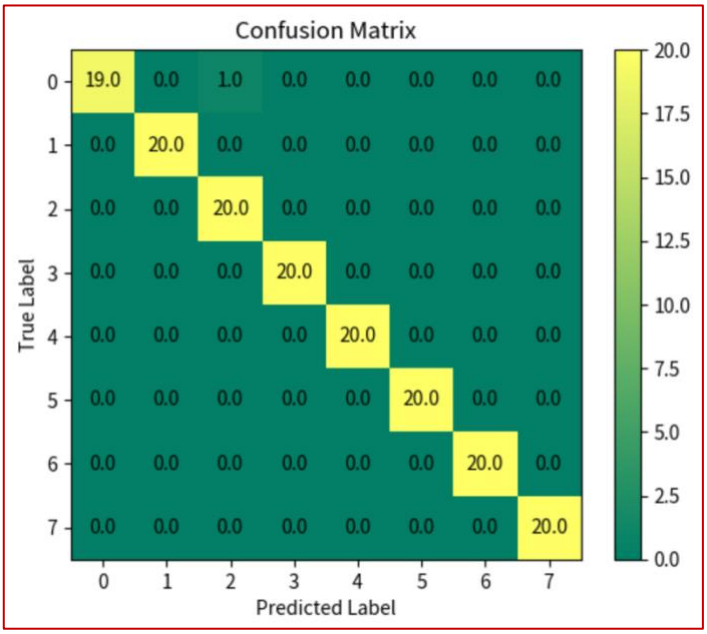
此外，GoogLeNet 還引入了全局平均池化層（Global Average Pooling），以減少參數量和計算量，從而降低了過度擬合的風險。

Structure:

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Result:

```
epoch: 0, Train loss: 0.2779, acc:0.6519,     Val loss: 0.0652, acc:0.9062
epoch: 1, Train loss: 0.1615, acc:0.8069,     Val loss: 0.0403, acc:0.9437
epoch: 2, Train loss: 0.1331, acc:0.8413,     Val loss: 0.0237, acc:0.9750
epoch: 3, Train loss: 0.1135, acc:0.8700,     Val loss: 0.0277, acc:0.9688
epoch: 4, Train loss: 0.0968, acc:0.8888,     Val loss: 0.0157, acc:0.9938
```



3. wide_resnet50_2

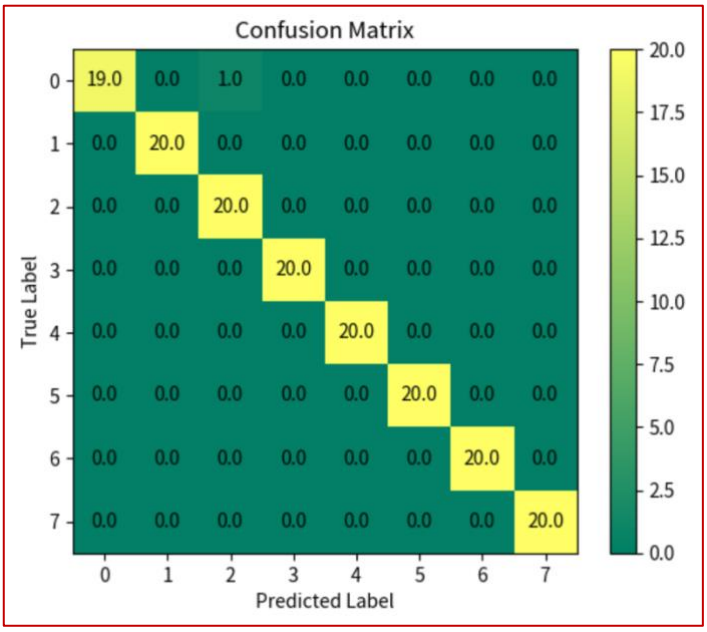
Introduction:

通過引入殘差連接來緩解深度神經網絡的訓練難度。這種結構有助於解決梯度消失和梯度爆炸等問題，使得模型更容易訓練。

由於 wide_resnet50_2 相對於標準的 ResNet 模型具有更大的模型容量，因此通常能夠在許多圖像識別任務中取得更好的性能。

Result:

```
epoch: 0, Train loss: 0.2528, acc:0.7025,     Val loss: 0.0733, acc:0.9625
epoch: 1, Train loss: 0.1264, acc:0.8475,     Val loss: 0.0319, acc:0.9875
epoch: 2, Train loss: 0.0931, acc:0.8856,     Val loss: 0.0174, acc:1.0000
epoch: 3, Train loss: 0.0670, acc:0.9162,     Val loss: 0.0173, acc:1.0000
epoch: 4, Train loss: 0.0636, acc:0.9187,     Val loss: 0.0083, acc:0.9938
```



4. regnet_y_400mf

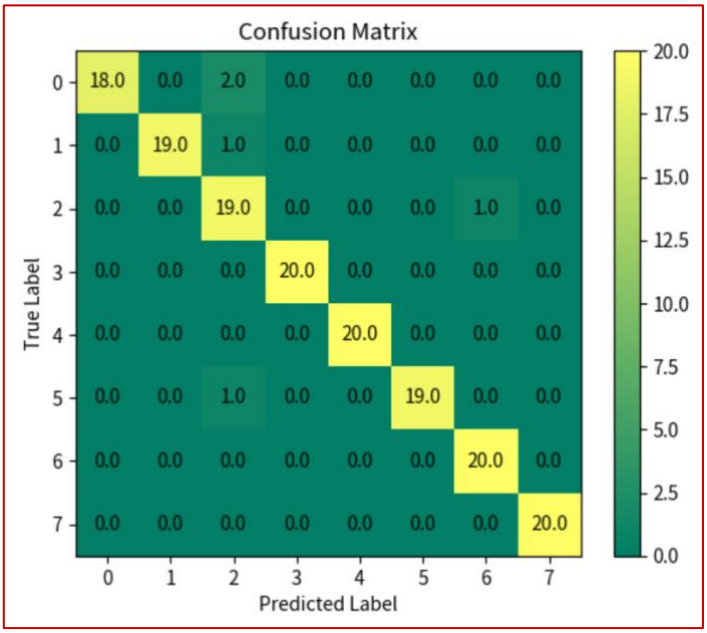
Introduction:

RegNet 是由 Facebook AI 提出的一種高效的神經網絡架構搜索方法，旨在提高計算效率和模型性能。

"y"表示模型的規模，而"400mf"代表了其在計算量上的規模。"mf"表示百萬浮點操作（million floating-point operations），而"400"則指的是模型在訓練時所需要的計算量，單位是百萬（million）浮點操作數。

Result:

```
epoch: 0, Train loss: 0.2592, acc:0.6731,     Val loss: 0.0997, acc:0.8812
epoch: 1, Train loss: 0.1444, acc:0.8169,     Val loss: 0.0512, acc:0.9625
epoch: 2, Train loss: 0.1234, acc:0.8456,     Val loss: 0.0417, acc:0.9500
epoch: 3, Train loss: 0.0927, acc:0.8856,     Val loss: 0.0210, acc:0.9938
epoch: 4, Train loss: 0.0838, acc:0.8956,     Val loss: 0.0270, acc:0.9688
```



5. resnext50_32x4d

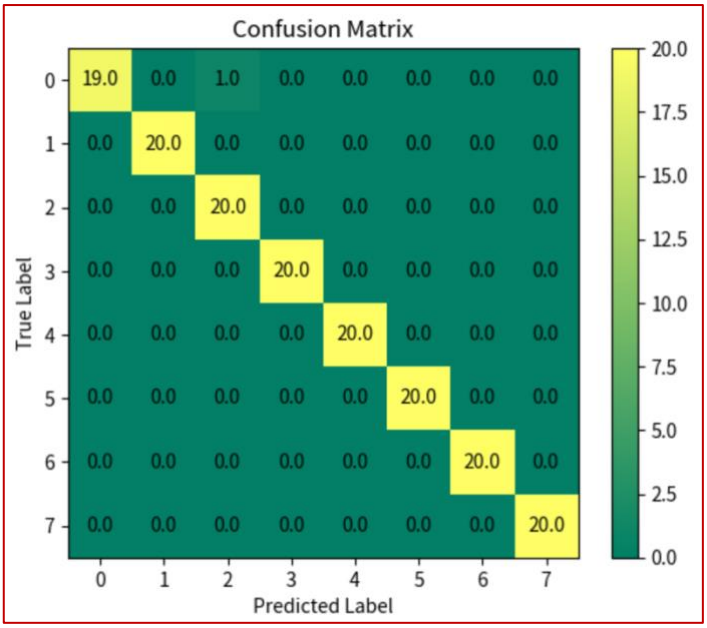
Introduction:

"50"代表模型的深度，即由 50 個卷積層組成。而"32x4d"則表示模型的基本單元是一個具有 32 個分支的組合單元（cardinality），每個分支都是一個 4 維的卷積核。這種設計使得模型在提取特徵時能夠更加充分和多樣化。

ResNeXt 的核心創新在於引入了所謂的“組卷積”（group convolution）的概念，即將輸入特徵圖分成多個組，每個組內進行獨立的卷積操作，然後將結果合併。這種結構可以在一定程度上提高模型的表示能力，同時又保持了計算效率。

Result:

```
epoch: 0, Train loss: 0.2631, acc:0.6687,     Val loss: 0.0742, acc:0.9563
epoch: 1, Train loss: 0.1393, acc:0.8338,     Val loss: 0.0445, acc:0.9688
epoch: 2, Train loss: 0.1053, acc:0.8737,     Val loss: 0.0224, acc:0.9750
epoch: 3, Train loss: 0.0872, acc:0.8888,     Val loss: 0.0169, acc:0.9875
epoch: 4, Train loss: 0.0671, acc:0.9237,     Val loss: 0.0146, acc:0.9938
```



6. shufflenet_v2_x0_5

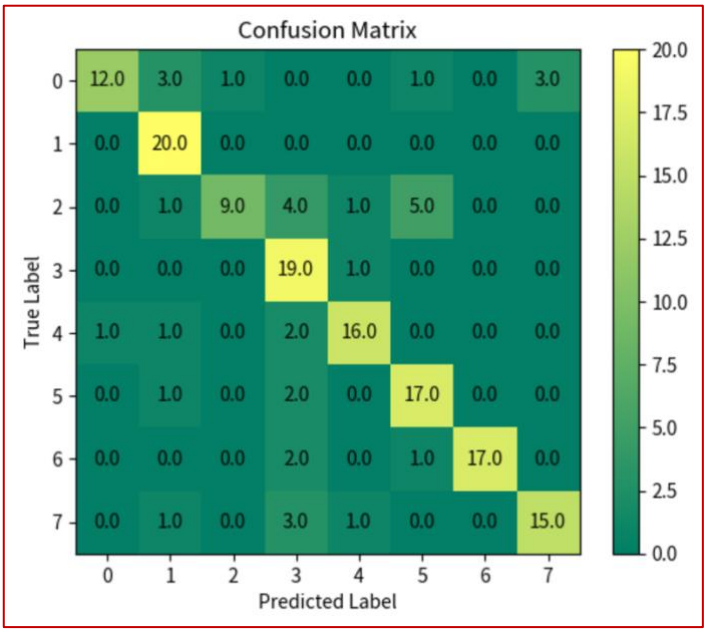
Introduction:

ShuffleNet V2_x0.5 的核心創新在於引入了深度可分離卷積和通道重排（channel shuffle）等技術，這些技術可以有效地減少模型的參數量和計算量，同時保持良好的性能。

深度可分離卷積將標準的卷積操作分為深度卷積和逐點卷積兩個步驟，從而減少了計算量。而通道重排則通過將輸入通道重新排列，使得不同通道的信息可以互相交換，增強了特徵提取的多樣性。

Result:

```
epoch: 0, Train loss: 0.5182, acc:0.2150,     Val loss: 0.5111, acc:0.6562
epoch: 1, Train loss: 0.5117, acc:0.5238,     Val loss: 0.4984, acc:0.7562
epoch: 2, Train loss: 0.5040, acc:0.5938,     Val loss: 0.4811, acc:0.7750
epoch: 3, Train loss: 0.4930, acc:0.6306,     Val loss: 0.4546, acc:0.7812
epoch: 4, Train loss: 0.4785, acc:0.6400,     Val loss: 0.4241, acc:0.7812
```



Conclusion

由於我選取的猴子的品種之間有顯著的特徵差異，因此無論使用哪一種的模型去做預測皆可以得到不錯的效果。

Callitrichinae

Rhinopithecus

Tarsiidae