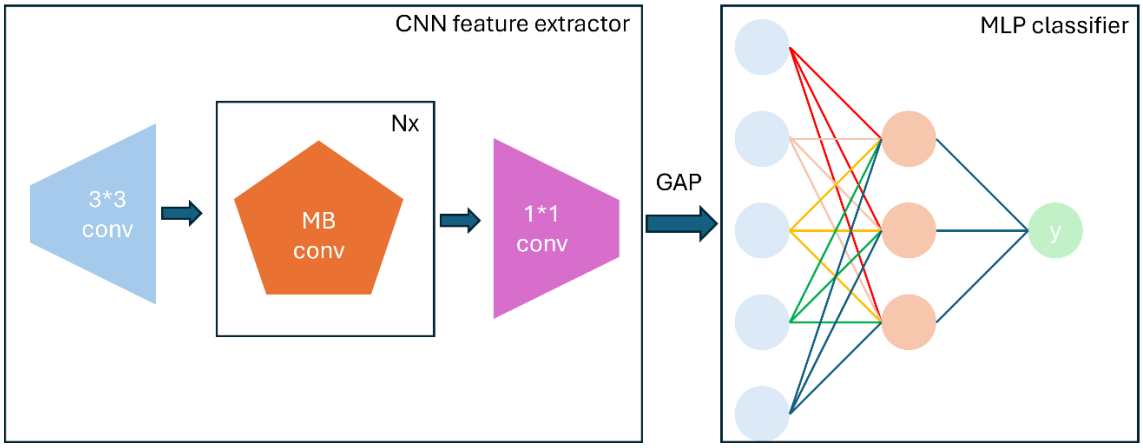
Machine learning final project

110511164 張語楹

Problem 1: How did you design your model for this final project?

本次期末專題會利用 convolution neural network (CNN) 來做 image classifier。CNN 的 image classifier 可以分成兩個部分，第一個是 feature extractor，另外一個是 classifier。而 feature extractor 可以當作是圖片的 encoder，通常是使用 CNN，而 classifier 用來輸出最後的機率分布，通常是 MLP。

這次專題我是使用 efficientnetv2 的架構作為 backbone，然後根據這個 backbone 來修改，efficientnetv2 使用了 MB convolution，下面部分會說明我更動 MB convolution 的地方和原因。



MB convolution:

Efficientnet 當中提到他所使用的 convolution 是 MB convolution。他有幾個特色:

1. Residual connection:這個在 Resnet 就有被使用，可以減少 gradient vanishing 的問題，增加訓練的穩定性。

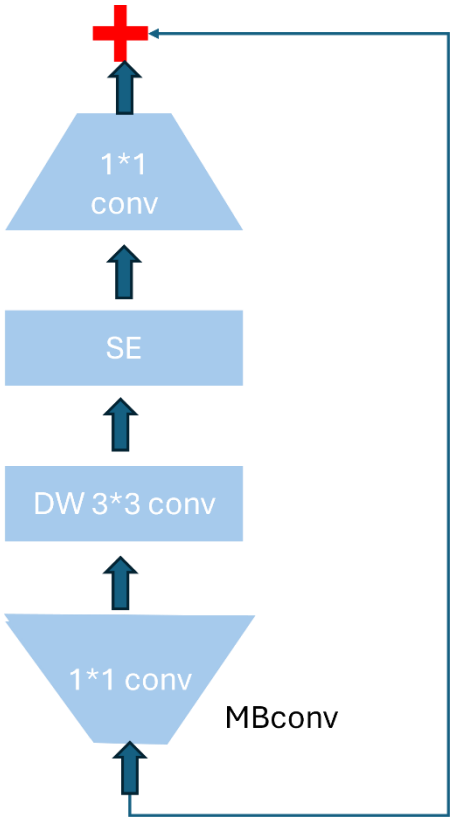
2. Conv1*1: 這個 layer 是為了增加 channel 的數量，在 Mobilenetv2 的論文當中有提到在 depthwise convolution 前要增加 channel 的數量才能避免資訊丟失。

3. Depthwise convolution: 這個 layer 就是最主要做 convolution 的 layer。其中採用 depthwise convolution 來減少參數量和算量。

4. SElayer: 用來加權 channel 的重要性。

5. Conv1*1: 這個 layer 是為了減少 channel 的數量，讓輸出的 channel 數量和輸入的 channel 數量一致，才能將輸入和輸出做加法 (residual connection)。

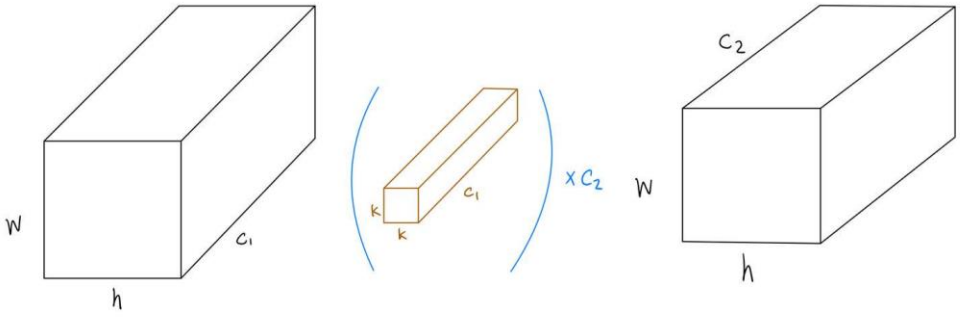
接下來，我會進一步分析 Depthwise convolution 和 SElayer:



Depth-wise-separable convolution:

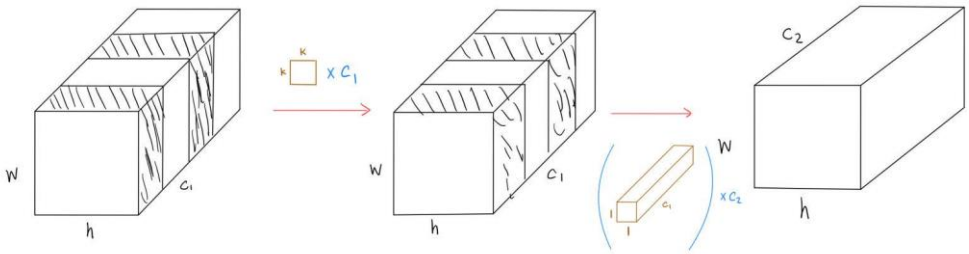
要減少算量和參數量最方便的方法就是把一般的 convolution 改成 depth-wise separable convolution。(假設 padding = 0, stride = 1, in_channel = c_1, out_channel = c_2, kernel_size = k, in_height = h, in_width = w)

一般的 convolution 如下圖:



FLOPS	Number of parameters
$(k^2 \cdot c_1 \cdot c_2) \cdot w \cdot h$	$k^2 \cdot c_1 \cdot c_2$

Depth-wise separable convolution 核心理念是將一般的 convolution 拆成兩個捲積。第一個 convolution 是在單一 channel 內的 feature map 先各自做 convolution。第二個 convolution 再做 1*1 的 convolution 融合 channel dimension 的資訊，並調整維度。



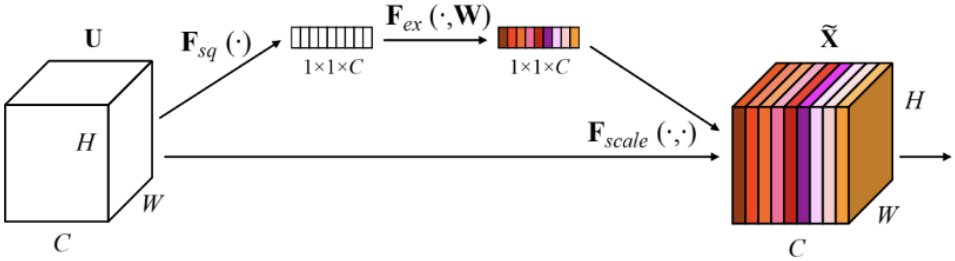
要分析 flops 和參數量可以由將兩步拆開後分析:

	FLOPS	Number of parameters
depthwise (第一步)	$(\mathrm{k}^2 \cdot \mathrm{c}_1) \cdot \mathrm{w} \cdot \mathrm{h}$	$k^2 \cdot c_1$
pointwise (第二步)	$(\mathrm{c}_1 \cdot \mathrm{c}_2) \cdot \mathrm{w} \cdot \mathrm{h}$	$\mathrm{c}_1 \cdot \mathrm{c}_2$
total	$\mathrm{c}_1 \cdot \mathrm{w} \cdot \mathrm{h}(\mathrm{k}^2 + \mathrm{c}_2)$	$c_1(k^2 + c_2)$

可以看到在算量和參數量方面，一般的 convolution 會和 k^2c_2 成正比，而 depth-wise-separable convolution 會和(k^2+c_2)成正比。也就是說，當c_2很大時，depth-wise-separable convolution 能夠得到很大的好處。

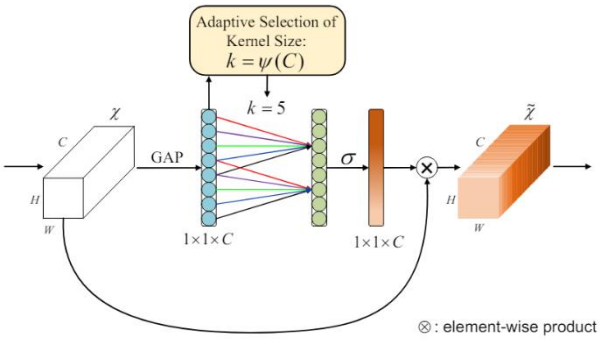
Squeeze and excitation layer (SElayer):

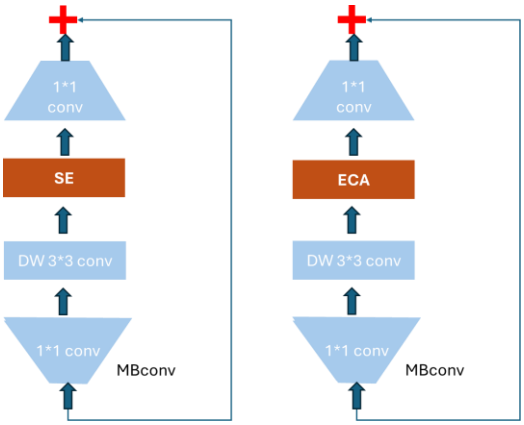
一張圖片經過 convolution 運算後，因為經過許多 kernel，所以在不同 channel 間會有不一樣的特徵被抓取出來，SElayer 是用來對每個 channel 做 weighted average。也就是說，他會對於不同的 feature map 去做不同重要性的評估。而那個 weight，就是使用 MLP 去計算。如此一來，模型就能夠判斷哪個 feature 是重要的 feature 並給予比較高的權重。所以它的用途是在評估每個 channel 間的重要性，因此輸入維度會等於輸出維度。



Efficient channel attention (ECA):

ECA 也是用來對每個 channel 做 weighted average。然而，ECA 並不是用 MLP 來計算權重，而是採用一維的 convolution 來計算權重。這個方法的優點是參數量可以大幅的減少，此外，算量也可以大幅的減少。然而，缺點就是一維的 convolution 的 kernel size 大大影響 channel 間資訊的傳遞。也就是說，一個 channel 只能看到 kernel size 的鄰居而已。本次專題為了減少算量和參數量，我將 **Squeeze and excitation layer** 換成 **Efficient channel attention**。

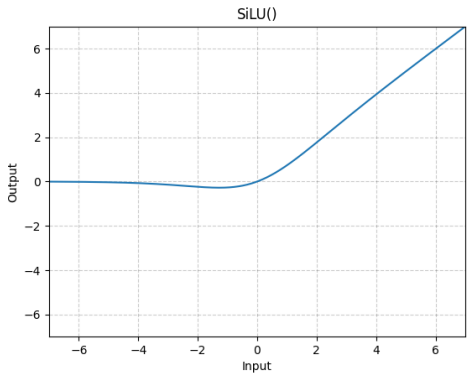




Activation function:

我所使用的 activation function 是 SiLU (Swish)

$$\text{silu}(x) = x * \sigma(x)$$



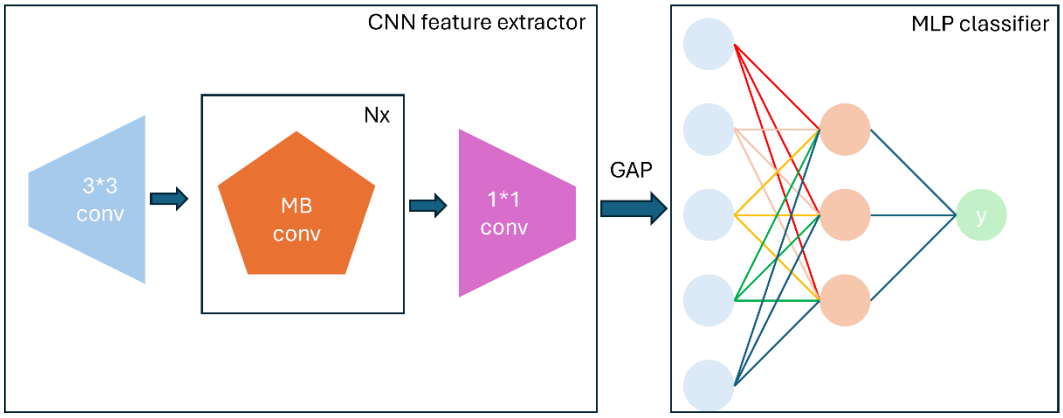
SiLU 和簡單的 ReLU 相當類似，但是他在負數的地方並不是全部都是零，且和 ReLU 在所有地方都可以微分。而現今許多模型都採用這個 activation function。

Why global average pooling (GAP)?

Global average pooling (GAP)非常好用。我在模型當中用了兩次 Global average pooling。

第一次是計算 Efficient channel attention (ECA)用來壓縮每個 channel 資訊的時候。第二次是在計算完 feature map 時利用 GAP 去壓縮每個 channel 的資訊，我如果沒有過 average pooling，那 feature extractor output dimension 會非常大，後面接上 MLP classifier 參數量和算量都會非常可觀。此外，加上 GAP 後我們就可以不管 feature map 的大小為多少，只要 channel 的數量正確就可以，加速建立模型的過程。

Model configuration:



MB convolution 是整個模型當中最重要的 building block。而 MB convolution 有五個參數可以調整。

Expansion ratio (t): expansion ratio 會決定在 MB convolution 內的 hidden dimension 是多少。

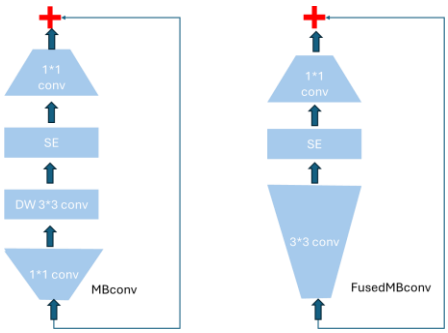
output channel (c): output channel 決定 MB convolution 的輸出的 channel dimension 是多少。

block repeat times(n): 決定該 MB convolution 有幾層。

stride (s): 決定 convolution 的 stride(步長)是多少。

enableFusedMB(EMB): 在 efficientnetv2 的論文當中提到在模型前幾層的 MB convolution 中的 depthwise separable convolution 會造成訓練速度變慢，因此論文提到將 MB convolution 換成 FusedMBconvolution，其實就是將 pointwise

convolution 和後面 3*3 的 depthwise separable convolution 換成一般的 3*3 convolution。



了解完參數的意義後，我有幾個模型設計上的考量:

首先，CNN 模型最重要的就是 receptive field，要增加 receptive field 有兩個方法，第一個是增加 kernel size，但會犧牲參數量和算量。第二個方法，就是增加模型的深度。因此，在選擇深度和寬度之間，我選擇增加深度而非增加寬度，也就是說，我會增加 MB conv 的數量而減少 convolution 的 channel 數量。此外，因為使用 fused-MB convolution 的考量是增加訓練的速度，而這不是本次專題的主要考量，因此全部都使用一般的 MB convolution。

Problem 2: Use images or tables to present the impact of training time, model complexity.

Wide v.s.Deep

configuration	FLOPS	parameters	Test Accuracy
[2, 8, 6, 1, 0] [2, 16, 4, 1, 0]	0.180348032G	0.010754M	0.7750
[2, 8, 4, 1, 0] [2, 16, 6, 1, 0]	0.212727168G	0.012738M	0.7667
[1, 8, 1, 1, 0] [1, 16, 1, 1, 0] [1, 32, 1, 1, 0] [1, 64, 1, 1, 0]	0.22129664G	0.014186M	0.7417

從上面表格可以看的出來，與其增加 channel 數量，不如增加模型的深度進而增加 receptive field。

Effect of expansion ratio (t):

Expansion ratio 影響 MB convolution 內的 hidden dimension 是多少。Expansion ratio 越大，模型不但 flops 增加，parameters 也增加不少，accuracy 也在預期內跟著上升。

configuration	FLOPS	parameters	Test Accuracy
[2, 8, 2, 1, 0], [2, 16, 2, 2, 0],	0.042099328G	0.005866M	0.7333
[4, 8, 2, 1, 0], [4, 16, 2, 2, 0],	0.068967168G	0.008826M	0.7417

Effect of stride (s):

Stride 會影響 convolution 的步長，當 stride 比較大的時候，output feature map 會比較小，所需要的算量也是一半，但經過實驗後發現 accuracy 會約略下降。

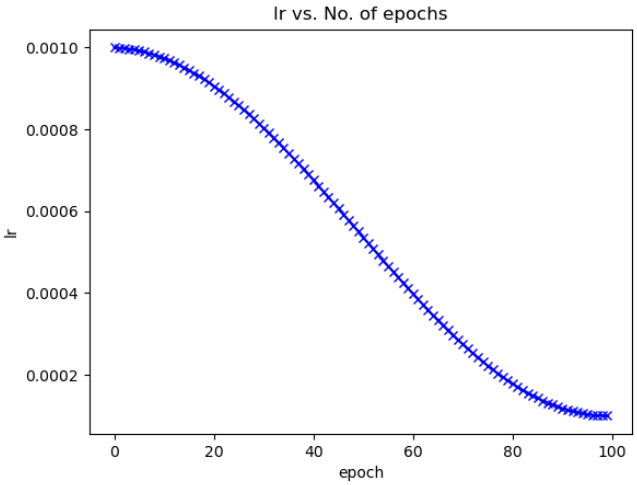
configuration	FLOPS	parameters	Test Accuracy
[2, 8, 2, 1, 0], [2, 16, 2, 1, 0],	0.096542848G	0.005866M	0.7333
[2, 8, 2, 1, 0], [2, 16, 2, 2, 0],	0.042099328G	0.005866M	0.7333
[2, 8, 2, 2, 0], [2, 16, 2, 2, 0],	0.015580288G	0.005866M	0.7167

結論來說，在相同的參數量下，模型深比寬好，此外，當模型越複雜時，準確率會有所提升。最極端的例子就是後面的 teacher model 達到 90%的準確度。

Problem 3: How did you train your model for this final project?

Learning rate schedular:

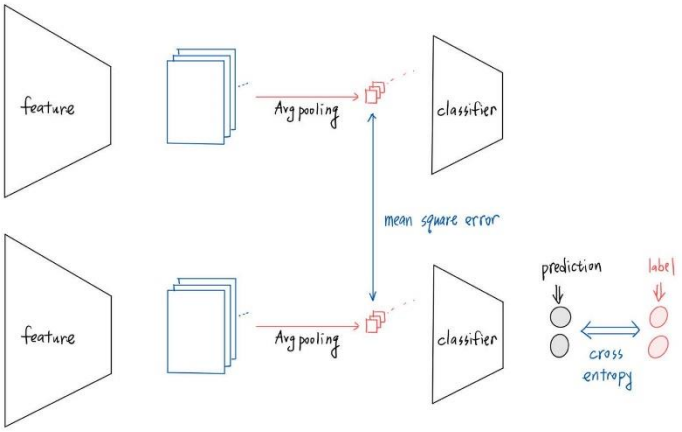
根據 HW2 的經驗，我在訓練的初期採用較高的 learning rate，之後慢慢地降低 learning rate。



Knowledge distillation:

這次的 final project 計算 loss 的方式不僅僅是使用模型輸出和 label 的 cross-entropy loss，而是另外訓練一個更大的模型，並盡可能讓小模型去模仿大模型的行為。

我的 loss 計算分為兩個部分。第一個部分是計算小模型的 feature extractor 和大模型的 feature extractor 的 mean square error。第二個部分是計算小模型的 prediction 和 ground truth 的 cross entropy loss。接下來將兩者做 weighted average。而那個 weighted 也是超參數的一環。

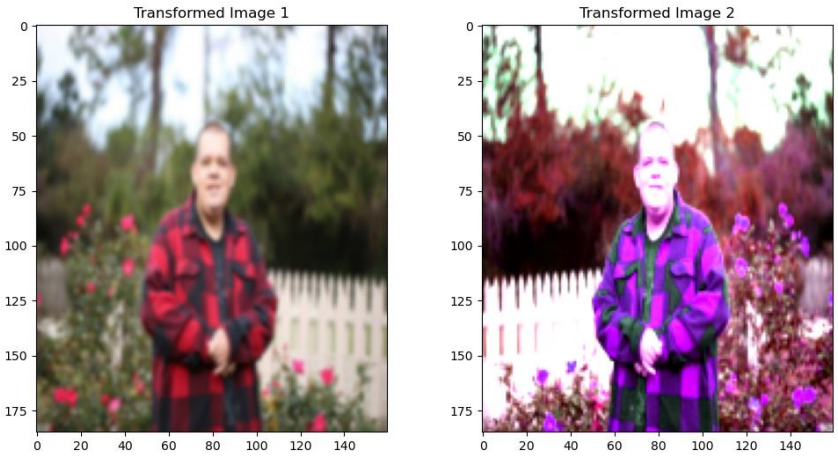


Teacher model:

FLOPS	Number of parameters	Accuracy
0.592416856G	5.495339M	90.7%

Data augmentation:

因為這次資料量較少，所以我利用 torchvision 一些功能來實現資料增強，包括 gaussian blur、horizontal flip、color jitter 等等。

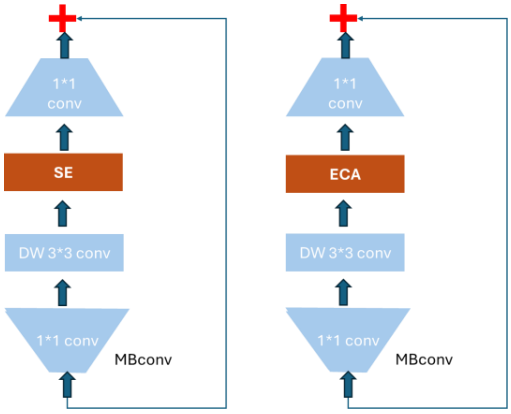


Problem 4 : Compare your proposed model with other methods.

我在 channel attention 的部分將 SElayer 換成 ECA，並發現模型的準確度有所提升，且算量和參數量都有所減少。

Squeeze and excitation layer (SElayer) v.s. **Efficient channel attention (ECA)**

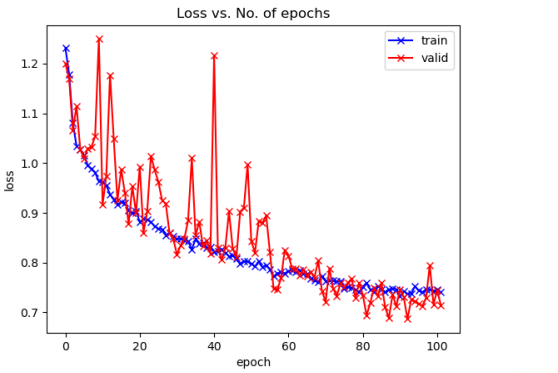
configuration	Channel attention	FLOPS	parameters	Test Accuracy
[2, 8, 6, 1, 0] [2, 16, 4, 1, 0]	ECA	0.180348032G	0.010754M	0.7750
[2, 8, 6, 1, 0] [2, 16, 4, 1, 0]	SE layer	0.18035344G	0.01433M	0.6917

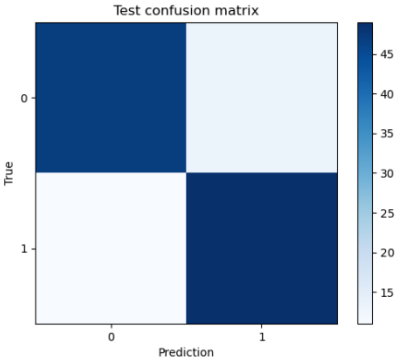


經過測試證實 ECA 確實夠以比較低的參數量和算量達到比 SE layer 更好的效果。因此本次專題將 SE layer 全部換成 ECA。

Problem 5 : Result.

configuration	Channel attention	FLOPS	parameters	Test Accuracy
[2, 8, 6, 1, 0] [2, 16, 4, 1, 0]	ECA	0.180348032G	0.010754M	0.7917





Github page: https://github.com/Benchangatrul284/NYCU_ML/tree/main/Final

Presentation video:

https://youtu.be/_7CGJ0pd2as