

人工智慧與影像辨識實務 HW4

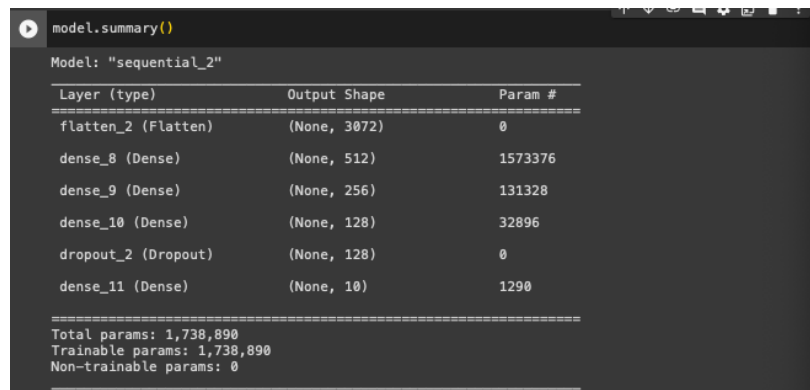
B10915030 陳奕軒

前言

為保持同樣基準點，每個 Model 都只訓練 10epochs。

第一題

About Model



```
model.summary()
```

Layer (type)	Output Shape	Param #
flatten_2 (Flatten)	(None, 3072)	0
dense_8 (Dense)	(None, 512)	1573376
dense_9 (Dense)	(None, 256)	131328
dense_10 (Dense)	(None, 128)	32896
dropout_2 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 10)	1290

Total params: 1,738,890
Trainable params: 1,738,890
Non-trainable params: 0

- 輸入處理
 - 將輸入扁平化： $32 * 32 * 3 \rightarrow 3072 * 1$
- 架構
 - 三層 Fully connection layer
 - ◆ $3072 \rightarrow 512$
 - ◆ $512 \rightarrow 256$
 - ◆ $256 \rightarrow 128$
- 輸出
 - 以 Dropout layer 丟棄 20%的神經元

- Fully connection layer(128->10) · activation 使用

softmax 輸出機率

- 其他資訊

- Optimizer 使用 Adam

- Loss 使用 Categorical_crossentropy

- Metric 使用 accuracy

比較

- 對象 : tf04_Cifar_Cnn2.ipynb

- Accuracy

題目 1

train accuracy: 0.49

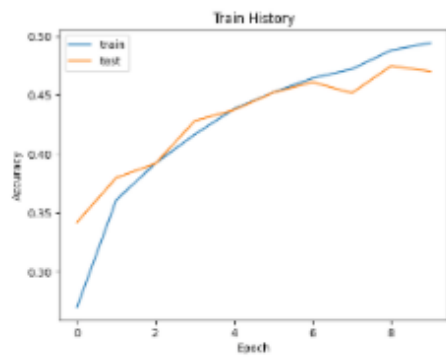
valid accuracy: 0.47

Tf04 勝

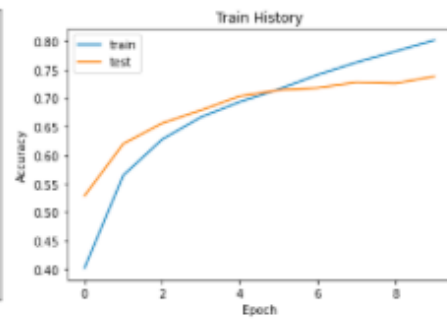
train accuracy: 0.8

valid accuracy: 0.47

題目1



tf04



- Loss

題目 1

Train loss 1.42

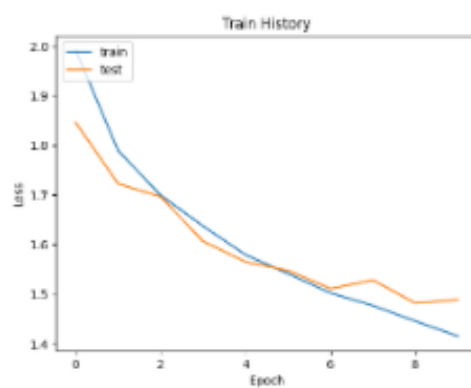
Validation loss 1.5

Tf04 勝

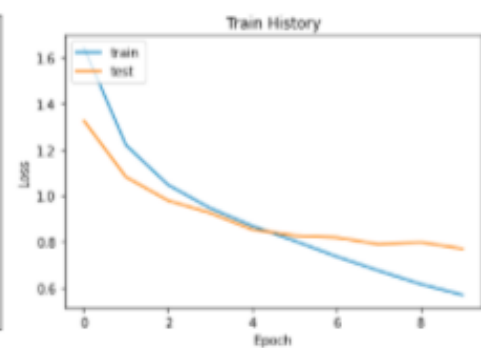
Train loss 0.6

Validation loss 0.8

題目1



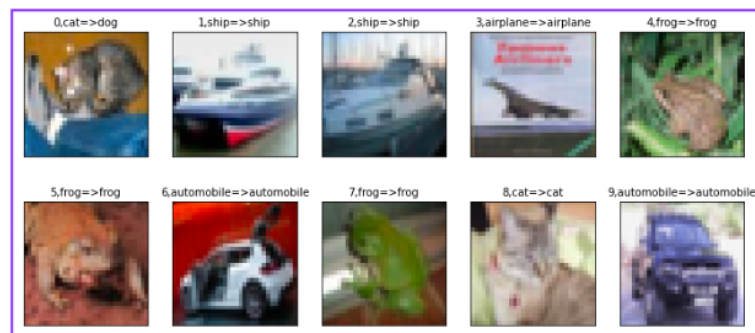
tf04



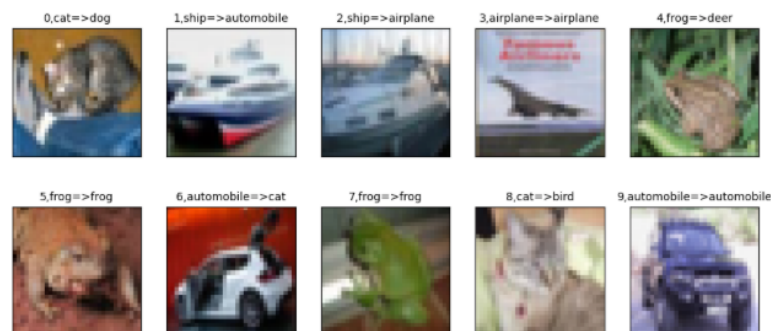
- 預測前 10 筆 (僅供參考)

題目 1 答對 4/10 , tf04 答對 9/10 , tf04 勝

tf04



題目1



- Confusion Matrix

題目 1 較多預測錯誤 , tf04 較少 , tf04 勝

tf04

predict	0	1	2	3	4	5	6	7	8	9
label										
0	667	32	109	13	16	9	46	8	61	39
1	15	820	9	7	3	4	50	7	17	68
2	60	4	511	40	72	84	191	20	10	8
3	8	10	119	313	55	193	271	18	4	9
4	24	5	154	35	422	35	278	38	9	0
5	6	4	115	106	48	544	143	28	2	4
6	0	2	33	24	12	9	916	2	0	2
7	11	1	64	37	83	114	67	610	0	13
8	106	75	37	20	5	11	33	2	682	29
9	41	183	12	15	6	13	76	18	26	610

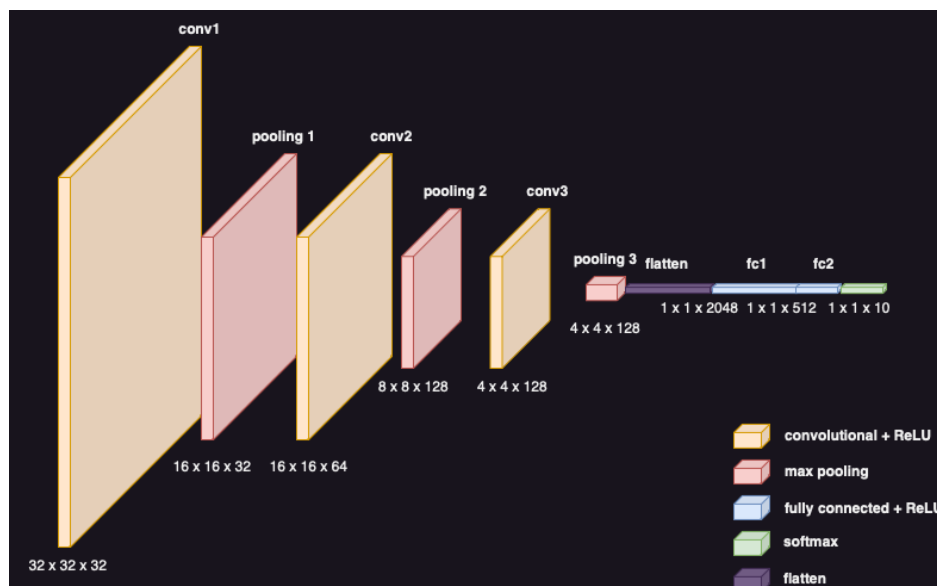
題目1

predict	0	1	2	3	4	5	6	7	8	9
label										
0	588	77	46	22	14	11	42	12	153	35
1	34	738	5	23	6	16	24	11	52	91
2	118	53	261	92	113	63	224	30	33	13
3	44	43	63	312	30	174	222	19	37	56
4	83	30	105	65	337	38	263	31	39	9
5	37	27	78	202	41	347	164	37	45	24
6	12	33	31	63	74	40	702	6	19	20
7	84	55	68	87	76	68	95	384	25	60
8	131	116	12	29	12	20	17	5	626	32
9	61	291	5	31	7	17	37	19	86	446

題目 2

Model Arichitecture

Layer (type)	Output Shape	Param #
conv2d_21 (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d_21 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_22 (Conv2D)	(None, 16, 16, 64)	18496
max_pooling2d_22 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_23 (Conv2D)	(None, 8, 8, 128)	204928
max_pooling2d_23 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_7 (Dropout)	(None, 4, 4, 128)	0
flatten_7 (Flatten)	(None, 2048)	0
dense_8 (Dense)	(None, 512)	1049088
dense_9 (Dense)	(None, 10)	5130
Total params: 1,278,538		
Trainable params: 1,278,538		
Non-trainable params: 0		



參數

1. Filter: 32, 64, 128
2. Kernel Size: (3x3), (3x3), (5x5)
3. Layer: 3
4. Stride: 1

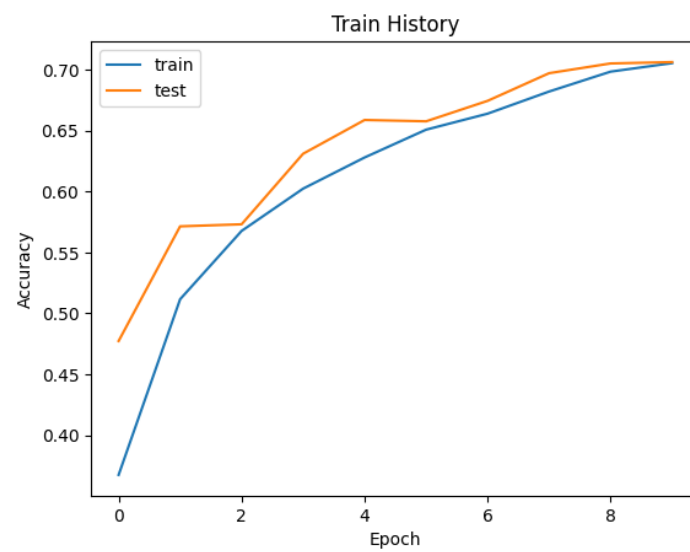
5. Batch size: 128
6. Dropout: 40%
7. Optimizer: Adam
8. Data Argument: 隨機水平 / 垂直翻轉

數據

- Accuracy

Train Accuracy: 0.7

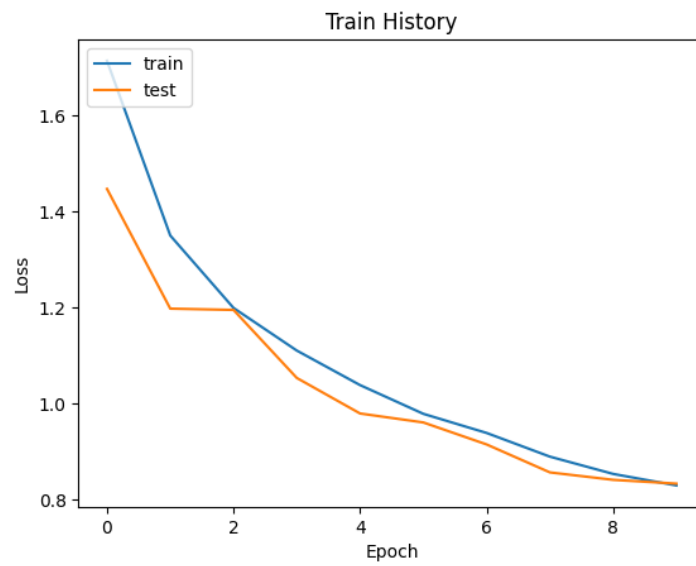
Validation Accuracy: 0.7



- Loss

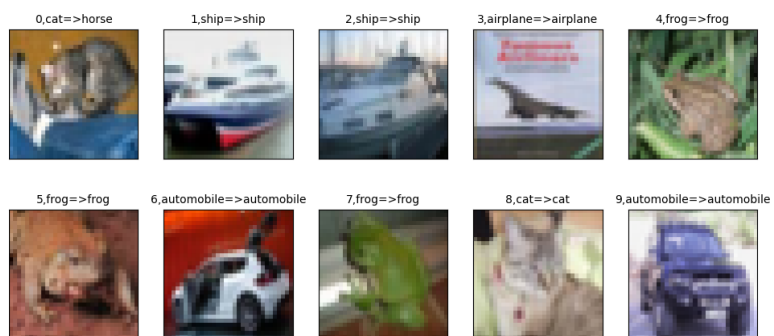
Train Loss: 0.83

Validation Loss: 0.83



- 預測前 10 筆 (僅供參考)

預測 9/10



- Confusion matrix

label										
0	723	27	50	32	18	7	11	18	69	45
1	6	838	2	19	6	3	12	5	7	102
2	59	5	443	86	201	64	103	13	12	14
3	13	15	38	492	102	190	84	40	8	18
4	13	3	37	45	787	22	27	53	9	4
5	5	7	27	168	72	629	31	43	4	14
6	4	1	22	52	79	28	801	3	4	6
7	10	6	23	62	100	51	11	722	0	15
8	40	70	20	10	19	16	9	6	767	43
9	18	80	3	19	7	9	6	11	19	828

評估

若觀察各種指標，可以說 tf04 和題目 2 的表現位於伯仲之間。在題目

2 中，我使用了一些特殊的參數，像是 Kernel size(5x5)、Dropout

40%、水平垂直翻轉等等，得到的結果卻和一般的 CNN 差不多。若要得

到更佳的结果可能要透過題目 3 的 Transfer Learning。

題目 3

網路架構

Model: "sequential_3"		
Layer (type)	Output Shape	Param #
=====		
mobilenetv2_1.00_96 (Functional)	(None, 3, 3, 1280)	2257984
sequential_1 (Sequential)	(None, 10)	165258
=====		
Total params: 2,423,242		
Trainable params: 2,026,698		
Non-trainable params: 396,544		
=====		

其中，Sequential 裡面的 Layer 如下圖：

```
[12] classification = model = tf.keras.models.Sequential([
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

使用的資料擴增方式和題目 2 一樣，兩者之間不同的地方在於題目 3

使用了學習率為 10^{-4} 的 adam，而題目 2 只使用了 10^{-3} 。此外，題目

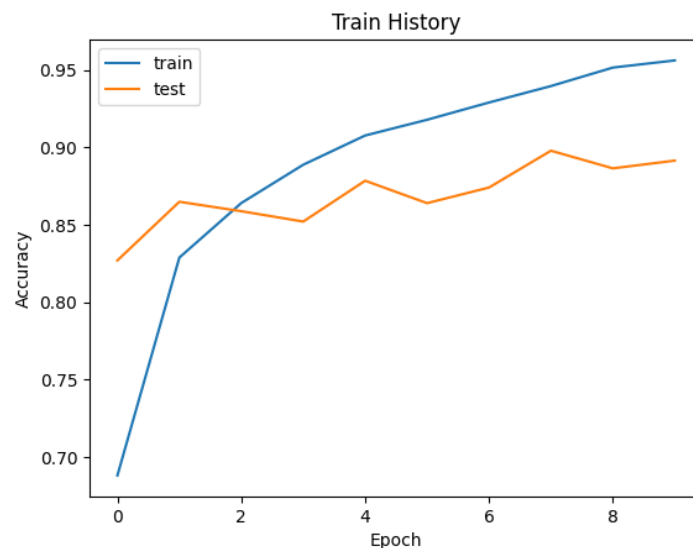
3 凍結了訓練好的 100 層，只訓練後來和新增的層數。

數據

- Accuracy

Train Accuracy: 0.956

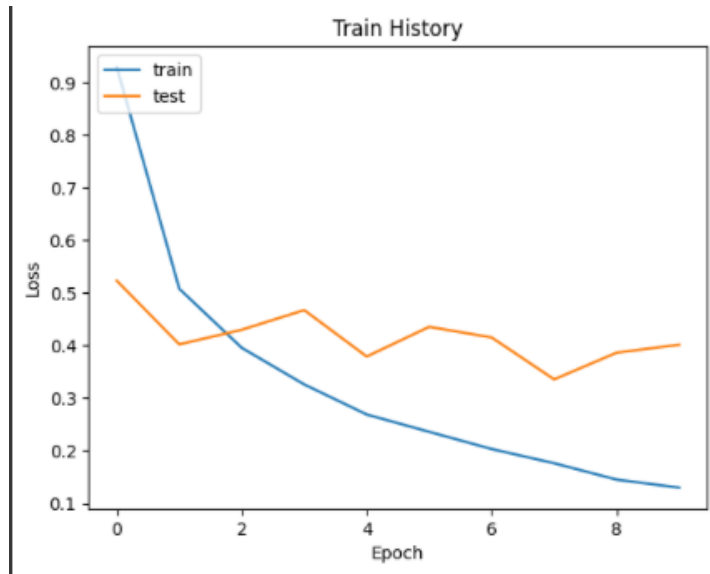
Validation Accuracy: 0.891



- Loss

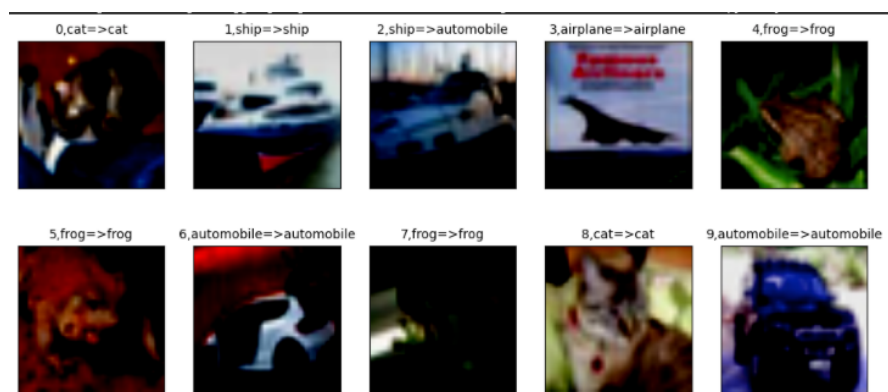
Train Loss: 0.13

Validation Loss: 0.4



- 預測前 10 筆 (僅供參考)

預測 9/10



- Confusion matrix

label										
0	885	18	13	5	6	1	7	5	49	11
1	2	979	0	0	0	0	2	0	6	11
2	19	2	835	39	20	16	63	3	3	0
3	4	9	9	777	17	94	70	5	11	4
4	6	3	30	27	802	29	46	51	4	2
5	4	3	9	104	10	829	27	11	2	1
6	3	1	1	6	2	4	982	0	1	0
7	8	2	10	16	18	29	12	897	5	3
8	18	15	1	2	0	0	5	0	957	2
9	5	104	0	3	0	0	3	0	17	868

評估

在題目 3 中我們可以發現，他的成果遠比題目 1、題目 2 和 tf04 還好上許多，就連 Confusion Matrix 都表現得比較集中。當然，這還是有最佳化空間的，不管是訓練多個 epoch、更換 pretrain model 或是調整分類層都有可能讓 Loss 進一步下降。

Model Colab Link

避免環境衝突

題目 1:

<https://colab.research.google.com/drive/1xED0ce6CWZPqEmW8WPa6kTO8jLQn66xl?usp=sharing>

題目 2:

<https://colab.research.google.com/drive/1J2TTpdLSkeAE0MHaUF8HyToZEGNdeRoM?usp=sharing>

題目 3:

https://colab.research.google.com/drive/1l-VXqo4v8LgUF4LnDLLOH7PgzeVu0_wb?usp=sharing