今天的主題是要探討優化器（Optimizer）對模型學習的影響，有關優化器該用哪個好，也是一個蠻令人頭痛的問題，大部分的時候優化器都可以讓你成功收斂，但有小部份時候優化器直接讓你訓練nan。

我們這次要比較的優化器從古早的SGD、Momentum、Adagrad、RMSProp、Adam，到較新的Range都有，要注意因為比較的優化器很多，很有可能會超出 Colab 使用時間上限，為了降低訓練時間，我們會做遷移式學習，鎖住模型142層以前的權重值，只專注訓後面的幾層作為觀察。

另外有關各個優化器的介紹，我在之前有寫過[一篇介紹文](https://ithelp.ithome.com.tw/articles/10221856)可以看看。

實驗一: SGD

全稱 Stochastic gradient descent，即最基本的 gradient。

base = tf.keras.applications.MobileNetV2(input\_shape=(224, 224, 3), include\_top=False, weights='imagenet')

net = tf.keras.layers.GlobalAveragePooling2D()(base.output)

net = tf.keras.layers.Dense(NUM\_OF\_CLASS)(net)

model = tf.keras.Model(inputs=[base.input], outputs=[net])

# Unfreeze weights

for idx, layer in enumerate(model.layers):

layer.trainable = FREEZE\_INDEX < idx

model.compile(

optimizer=tf.keras.optimizers.SGD(LR),

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=[tf.keras.metrics.SparseCategoricalAccuracy()],

)

start = timeit.default\_timer()

sgd\_history = model.fit(

ds\_train,

epochs=EPOCHS,

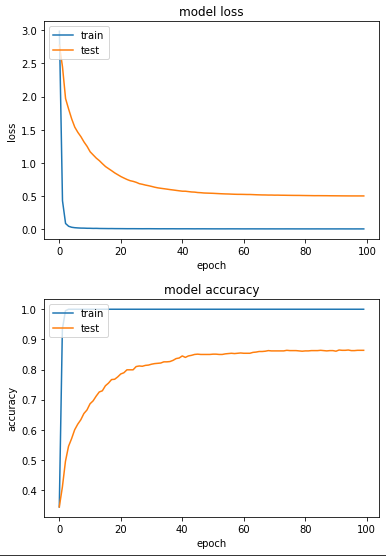
validation\_data=ds\_test,

verbose=True)

print(f'cost {timeit.default\_timer()-start} sec')

產出:

loss: 0.0011 - sparse\_categorical\_accuracy: 1.0000 - val\_loss: 0.4991 - val\_sparse\_categorical\_accuracy: 0.8637



實驗二：Momentum

在SGD中多加了動量的概念。

base = tf.keras.applications.MobileNetV2(input\_shape=(224, 224, 3), include\_top=False, weights='imagenet')

net = tf.keras.layers.GlobalAveragePooling2D()(base.output)

net = tf.keras.layers.Dense(NUM\_OF\_CLASS)(net)

model = tf.keras.Model(inputs=[base.input], outputs=[net])

# Unfreeze weights

for idx, layer in enumerate(model.layers):

layer.trainable = FREEZE\_INDEX < idx

model.compile(

optimizer=tf.keras.optimizers.SGD(LR, momentum=0.9),

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=[tf.keras.metrics.SparseCategoricalAccuracy()],

)

start = timeit.default\_timer()

mom\_history = model.fit(

ds\_train,

epochs=EPOCHS,

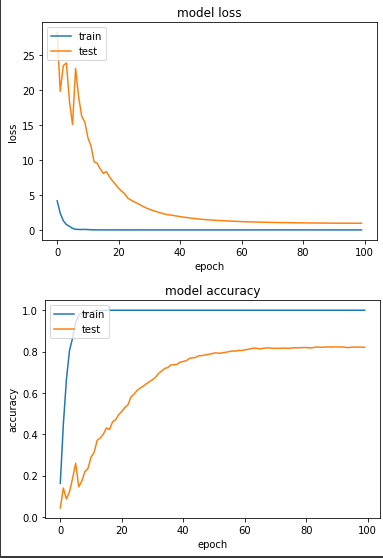
validation\_data=ds\_test,

verbose=True)

print(f'cost {timeit.default\_timer()-start} sec')

產出：

loss: 9.9336e-05 - sparse\_categorical\_accuracy: 1.0000 - val\_loss: 0.9496 - val\_sparse\_categorical\_accuracy: 0.8206



實驗三：Adagrad

在SGD多加了快取的概念

base = tf.keras.applications.MobileNetV2(input\_shape=(224, 224, 3), include\_top=False, weights='imagenet')

net = tf.keras.layers.GlobalAveragePooling2D()(base.output)

net = tf.keras.layers.Dense(NUM\_OF\_CLASS)(net)

model = tf.keras.Model(inputs=[base.input], outputs=[net])

# Unfreeze weights

for idx, layer in enumerate(model.layers):

layer.trainable = FREEZE\_INDEX < idx

model.compile(

optimizer=tf.keras.optimizers.Adagrad(LR, initial\_accumulator\_value=0.1),

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=[tf.keras.metrics.SparseCategoricalAccuracy()],

)

start = timeit.default\_timer()

ada\_history = model.fit(

ds\_train,

epochs=EPOCHS,

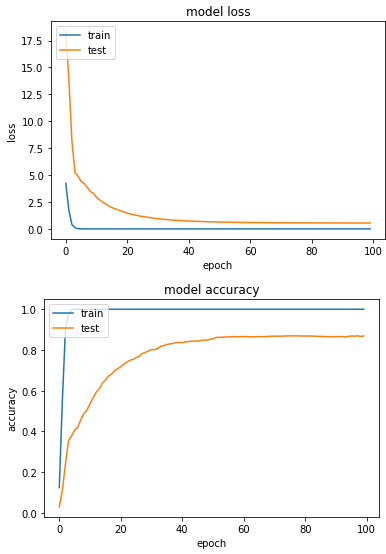
validation\_data=ds\_test,

verbose=True)

print(f'cost {timeit.default\_timer()-start} sec')

產出：

loss: 2.8722e-04 - sparse\_categorical\_accuracy: 1.0000 - val\_loss: 0.5482 - val\_sparse\_categorical\_accuracy: 0.8686



實驗四：RMSProp

在 Adagrad 中多加了 decay 的概念。這邊由於我自己測試時，發現LR=0.1時，模型非常不穩定，所以此處LR改成0.001。

base = tf.keras.applications.MobileNetV2(input\_shape=(224, 224, 3), include\_top=False, weights='imagenet')

net = tf.keras.layers.GlobalAveragePooling2D()(base.output)

net = tf.keras.layers.Dense(NUM\_OF\_CLASS)(net)

model = tf.keras.Model(inputs=[base.input], outputs=[net])

# Unfreeze weights

for idx, layer in enumerate(model.layers):

layer.trainable = FREEZE\_INDEX < idx

model.compile(

optimizer=tf.keras.optimizers.RMSprop(0.001, rho=0.99),

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=[tf.keras.metrics.SparseCategoricalAccuracy()],

)

start = timeit.default\_timer()

rms\_history = model.fit(

ds\_train,

epochs=EPOCHS,

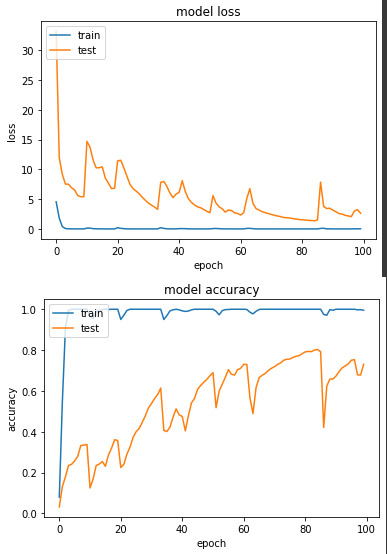
validation\_data=ds\_test,

verbose=True)

print(f'cost {timeit.default\_timer()-start} sec')

產出:

loss: 0.0232 - sparse\_categorical\_accuracy: 0.9951 - val\_loss: 2.6411 - val\_sparse\_categorical\_accuracy: 0.7304



圖表上產生了有鋸齒狀的線，我認為應該是模型仍在多個 local minima 跳躍。

實驗五：Adam

帶入mean和var兩個概念。同樣發現LR=0.1時，模型不穩定，LR改成0.001。

base = tf.keras.applications.MobileNetV2(input\_shape=(224, 224, 3), include\_top=False, weights='imagenet')

net = tf.keras.layers.GlobalAveragePooling2D()(base.output)

net = tf.keras.layers.Dense(NUM\_OF\_CLASS)(net)

model = tf.keras.Model(inputs=[base.input], outputs=[net])

# Unfreeze weights

for idx, layer in enumerate(model.layers):

layer.trainable = FREEZE\_INDEX < idx

model.compile(

optimizer=tf.keras.optimizers.Adam(0.001, beta\_1=0.9, beta\_2=0.999),

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=[tf.keras.metrics.SparseCategoricalAccuracy()],

)

start = timeit.default\_timer()

adam\_history = model.fit(

ds\_train,

epochs=EPOCHS,

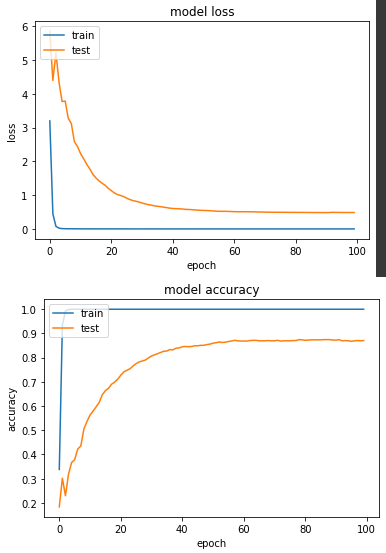
validation\_data=ds\_test,

verbose=True)

print(f'cost {timeit.default\_timer()-start} sec')

產出：

loss: 7.5301e-05 - sparse\_categorical\_accuracy: 1.0000 - val\_loss: 0.4853 - val\_sparse\_categorical\_accuracy: 0.8706



第六個實驗：Ranger

這個比較特別，這是一個結合RAdam和LookAhead(另外兩個新型優化器)的優化器，[原作Repo](https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer" \t "_blank)

只是這東西目前要使用的話，用 tensorflow addons 會比較方便。

!pip install -U tensorflow-addons

import tensorflow\_addons as tfa

一樣測試後，發現LR=0.001比較正常。

base = tf.keras.applications.MobileNetV2(input\_shape=(224, 224, 3), include\_top=False, weights='imagenet')

net = tf.keras.layers.GlobalAveragePooling2D()(base.output)

net = tf.keras.layers.Dense(NUM\_OF\_CLASS)(net)

model = tf.keras.Model(inputs=[base.input], outputs=[net])

# Unfreeze weights

for idx, layer in enumerate(model.layers):

layer.trainable = FREEZE\_INDEX < idx

radam = tfa.optimizers.RectifiedAdam(0.001)

ranger = tfa.optimizers.Lookahead(radam, sync\_period=6, slow\_step\_size=0.5)

model.compile(

optimizer=ranger,

loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

metrics=[tf.keras.metrics.SparseCategoricalAccuracy()],

)

start = timeit.default\_timer()

range\_history = model.fit(

ds\_train,

epochs=100,

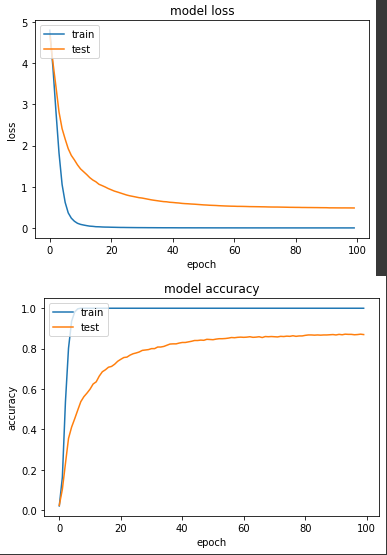
validation\_data=ds\_test,

verbose=True)

print(f'cost {timeit.default\_timer()-start} sec')

產出：

loss: 4.8191e-04 - sparse\_categorical\_accuracy: 1.0000 - val\_loss: 0.4846 - val\_sparse\_categorical\_accuracy: 0.8696



以上就是我們針對六種不同的優化器訓練同一個模型的實驗，以我自己實務經驗，我其實也是個跟風仔，會先嘗試使用比較新型的優化器，但如果訓練過程中發生 loss 不斷增大的狀況，我會再切成 SGD 來 debug 模型或調整 learning rate 來檢查有沒有問題。