TensorFlow2教程-keras模型保存和序列化

1 保存序列模型或函數式模型

In [1]:
```
# 構建一個簡單的模型並訓練
from __future__ import absolute_import, division, print_function
import tensorflow as tf
tf.keras.backend.clear_session()
from tensorflow import keras
from tensorflow.keras import layers

inputs = keras.Input(shape=(784,), name='digits')
x = layers.Dense(64, activation='relu', name='dense_1')(inputs)
x = layers.Dense(64, activation='relu', name='dense_2')(x)
outputs = layers.Dense(10, activation='softmax', name='predictions')(x)

model = keras.Model(inputs=inputs, outputs=outputs, name='3_layer_mlp')
model.summary()
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
x_train = x_train.reshape(60000, 784).astype('float32') / 255
x_test = x_test.reshape(10000, 784).astype('float32') / 255

model.compile(loss='sparse_categorical_crossentropy',
              optimizer=keras.optimizers.RMSprop())
history = model.fit(x_train, y_train,
                    batch_size=64,
                    epochs=1)

predictions = model.predict(x_test)
```

```
Model: "3_layer_mlp"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
digits (InputLayer)          [(None, 784)]             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                50240
_________________________________________________________________
dense_2 (Dense)              (None, 64)                4160
_________________________________________________________________
predictions (Dense)          (None, 10)                650
=================================================================
Total params: 55,050
Trainable params: 55,050
Non-trainable params: 0
_________________________________________________________________
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datase
ts/mnist.npz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnis
t.npz)
11493376/11490434 [==============================] - 1s 0us/step
938/938 [==============================] - 2s 2ms/step - loss: 0.3147
```

1.1 保存整個模型

可以對整個模型進行保存，其保持的內容包括：

- 該模型的架構
- 模型的權重（在訓練期間學到的）
- 模型的訓練配置（傳遞給編譯的）
- 優化器及其狀態（這使您可以從中斷的地方重新啟動訓練）

In [2]:
```
import numpy as np
# 模型保存
model.save('the_save_model.h5')
# 導入模型
new_model = keras.models.load_model('the_save_model.h5')
new_prediction = new_model.predict(x_test)
np.testing.assert_allclose(predictions, new_prediction, atol=1e-6) # 預測結果一樣
```

1.2 匯出為SavedModel文件

SavedModel是Tensorflow物件的獨立序列化格式，支援使用Tensorflow Serving server來部署模型，支援其他語言讀取。

In [3]:
```
# 匯出為tf的SavedModel文件
model.save('save_model', save_format='tf')
# 從SavedModel檔中導入模型
new_model = keras.models.load_model('save_model')

new_prediction = new_model.predict(x_test)
np.testing.assert_allclose(predictions, new_prediction, atol=1e-6) # 預測結果一樣
```

```
WARNING:tensorflow:From c:\users\asus_ux331\appdata\local\programs\python\pytho
n37\lib\site-packages\tensorflow\python\training\tracking\tracking.py:111: Mode
l.state_updates (from tensorflow.python.keras.engine.training) is deprecated an
d will be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied auto
matically.
WARNING:tensorflow:From c:\users\asus_ux331\appdata\local\programs\python\pytho
n37\lib\site-packages\tensorflow\python\training\tracking\tracking.py:111: Laye
r.updates (from tensorflow.python.keras.engine.base_layer) is deprecated and wi
ll be removed in a future version.
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied auto
matically.
INFO:tensorflow:Assets written to: save_model\assets
```

SaveModel創建的檔包含：

- 權重

- 網路圖

1.3 僅保存網路結構

僅保持網路結構，這樣匯出的模型並未包含訓練好的參數

In [5]:
```
# 獲取網路結構配置
config = model.get_config()
reinitialized_model = keras.Model.from_config(config)

new_prediction = reinitialized_model.predict(x_test)
assert abs(np.sum(predictions-new_prediction)) >0
```

也可以使用json保存網路結構

In [6]:
```
# 將網路結構匯出為json格式
json_config = model.to_json()
reinitialized_model = keras.models.model_from_json(json_config)

new_prediction = reinitialized_model.predict(x_test)
assert abs(np.sum(predictions-new_prediction)) >0
```

1.4 僅保存網路權重參數

In [4]:
```
# 獲取網路權重
weights = model.get_weights()
# 對網路權重進行賦值
model.set_weights(weights)
```

可以把結構和參數保存結合起來

In [5]:
```
config = model.get_config()
weights = model.get_weights()

new_model = keras.Model.from_config(config) # config只能用keras.Model的这个api
new_model.set_weights(weights)

new_predictions = new_model.predict(x_test)
np.testing.assert_allclose(predictions, new_predictions, atol=1e-6)
```

1.5 完整的模型保存方法

In [6]:
```
# 匯出網路結構和權重
json_config = model.to_json()
with open('model_config.json', 'w') as json_file:
    json_file.write(json_config)
model.save_weights('path_to_my_weights.h5')
# 載入網路結構和權重
with open('model_config.json') as json_file:
    json_config = json_file.read()
new_model = keras.models.model_from_json(json_config)
new_model.load_weights('path_to_my_weights.h5')

new_predictions = new_model.predict(x_test)
np.testing.assert_allclose(predictions, new_predictions, atol=1e-6)
```

In [11]:
```
# 當然也可以一步到位
model.save('path_to_my_model.h5')
del model
model = keras.models.load_model('path_to_my_model.h5')
```

```
W1013 16:49:21.690537 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer
W1013 16:49:21.691003 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer.iter
W1013 16:49:21.691583 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer.decay
W1013 16:49:21.691886 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer.learning_rate
W1013 16:49:21.692354 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer.momentum
W1013 16:49:21.692843 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer.rho
W1013 16:49:21.693265 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer's state 'rms' for (root).layer_with_weights-0.kernel
W1013 16:49:21.693663 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer's state 'rms' for (root).layer_with_weights-0.bias
W1013 16:49:21.693997 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer's state 'rms' for (root).layer_with_weights-1.kernel
W1013 16:49:21.694370 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer's state 'rms' for (root).layer_with_weights-1.bias
W1013 16:49:21.694779 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer's state 'rms' for (root).layer_with_weights-2.kernel
W1013 16:49:21.695171 140085562951488 util.py:144] Unresolved object in checkpo
int: (root).optimizer's state 'rms' for (root).layer_with_weights-2.bias
W1013 16:49:21.695466 140085562951488 util.py:152] A checkpoint was restored
(e.g. tf.train.Checkpoint.restore or tf.keras.Model.load_weights) but not all c
heckpointed values were used. See above for specific issues. Use expect_partial
() on the load status object, e.g. tf.train.Checkpoint.restore(...).expect_part
ial(), to silence these warnings, or use assert_consumed() to make the check ex
plicit. See https://www.tensorflow.org/alpha/guide/checkpoints#loading_mechanic
s (https://www.tensorflow.org/alpha/guide/checkpoints#loading_mechanics) for de
tails.
```

1.6 權重保存格式

有.h5或.keras尾碼時保存為keras HDF5格式檔，否則默認為TensorFlow Checkpoint格式檔。可以使用save_format顯式確定。

In [12]:
```
model.save_weights('weight_tf_savedmodel')
model.save_weights('weight_tf_savedmodel.h5')
```

In [13]:
```
model.save_weights('weight_tf_savedmodel_tf', save_format='tf')
model.save_weights('weight_tf_savedmodel_h5', save_format='h5')
```

1.7 子類模型權重保存

子類模型的結構無法保存和序列化，只能保持參數

In [16]:
```
# 構建模型
class ThreeLayerMLP(keras.Model):

    def __init__(self, name=None):
        super(ThreeLayerMLP, self).__init__(name=name)
        self.dense_1 = layers.Dense(64, activation='relu', name='dense_1')
        self.dense_2 = layers.Dense(64, activation='relu', name='dense_2')
        self.pred_layer = layers.Dense(10, activation='softmax', name='prediction

    def call(self, inputs):
        x = self.dense_1(inputs)
        x = self.dense_2(x)
        return self.pred_layer(x)

def get_model():
    return ThreeLayerMLP(name='3_layer_mlp')

model = get_model()
```

首先，無法保存從未使用過的子類模型。

這是因為需要在某些資料上調用子類模型才能創建其權重。

In [17]:
```
# 訓練模型
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
x_train = x_train.reshape(60000, 784).astype('float32') / 255
x_test = x_test.reshape(10000, 784).astype('float32') / 255

model.compile(loss='sparse_categorical_crossentropy',
              optimizer=keras.optimizers.RMSprop())
history = model.fit(x_train, y_train,
                    batch_size=64,
                    epochs=1)
```

```
Train on 60000 samples
60000/60000 [==============================] - 2s 26us/sample - loss: 0.3128
```

推薦的保存子類模型的方法是使用save_weights創建TensorFlow SavedModel檢查點。

該檢查點將包含與模型關聯的所有變數的值：

- 圖層的權重
- 優化器的狀態
- 與有狀態模型指標關聯的任何變數

In [18]:
```
# 保持權重參數
model.save_weights('my_model_weights', save_format='tf')

# 輸出結果，供後面對比

predictions = model.predict(x_test)
first_batch_loss = model.train_on_batch(x_train[:64], y_train[:64])
```

要還原模型，將需要訪問創建模型物件的代碼。 請注意，為了恢復優化器狀態和任何有狀態度量的狀態，應該先編譯模型（使用與以前完全相同的參數）。

In [19]:
```
# 讀取保存的模型參數
new_model = get_model()
new_model.compile(loss='sparse_categorical_crossentropy',
                  optimizer=keras.optimizers.RMSprop())

#new_model.train_on_batch(x_train[:1], y_train[:1])

new_model.load_weights('my_model_weights')

new_predictions = new_model.predict(x_test)
np.testing.assert_allclose(predictions, new_predictions, atol=1e-6)


new_first_batch_loss = new_model.train_on_batch(x_train[:64], y_train[:64])
assert first_batch_loss == new_first_batch_loss
```

In [ ]:
```

```