Save and restore models

June 16, 2019

1 Save and restore models

Model progress can be saved during — and after — training. It is good practice to share: * code to create a model * the trained weights and parameters

How to save a model depends on the API used

[1]:
```
from __future__ import absolute_import, division, print_function

import os

import tensorflow as tf
from tensorflow import keras

tf.__version__
```

[1]:
```
'1.13.1'
```

Loading data Here the simple and well-known dataset MNIST is used, since the goal is to explore the saving and restoring mechanisms.

[2]:
```
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.
 ↪mnist.load_data()

train_labels = train_labels[:1000]
test_labels = test_labels[:1000]

train_images = train_images[:1000].reshape(-1, 28 * 28) / 255.0 # -1 infers the␣
 ↪shape => 1000 in this case (nb of examples)
test_images = test_images[:1000].reshape(-1, 28 * 28) / 255.0
```

Building a simple model This model uses sparse categorical crosentropy, because the target labels are integers. If we used categorical crossentropy we would need to encode them with one-hot.

[17]:
```
# Returns a short sequential model
def create_model():
  model = tf.keras.models.Sequential([
    keras.layers.Dense(512, activation=tf.keras.activations.relu,␣
 ↪input_shape=(784,)),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(10, activation=tf.keras.activations.softmax)
  ])

  model.compile(optimizer='adam',
                loss='sparse_categorical_crossentropy',
                metrics=['accuracy'])

  return model


# Create a basic model instance
model = create_model()
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_16 (Dense)             (None, 512)               401920
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_17 (Dense)             (None, 10)                5130
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

1.1 Save checkpoints during training

The primary use case is to automatically save checkpoints during and at the end of training. This way we can use a trained model without having to retrain it, or pick-up training where you left of—in case the training process was interrupted.

[18]:
```
checkpoint_path = "training_1/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

# Create checkpoint callback
cp_callback = tf.keras.callbacks.ModelCheckpoint(checkpoint_path,
                                                 save_weights_only=True,
                                                 verbose=1)

model = create_model()

model.fit(train_images, train_labels,  epochs = 10,
          validation_data = (test_images,test_labels),
          callbacks = [cp_callback])  # pass callback to training

# This may generate warnings related to saving the state of the optimizer.
# These warnings (and similar warnings throughout this notebook)
# are in place to discourage outdated usage, and can be ignored.
```

```
Train on 1000 samples, validate on 1000 samples
Epoch 1/10
 960/1000 [===========================>..] - ETA: 0s - loss: 1.1548 - acc:
0.6833
Epoch 00001: saving model to training_1/cp.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A09A90EB8>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
1000/1000 [==============================] - 1s 922us/sample - loss: 1.1396 -
acc: 0.6850 - val_loss: 0.6710 - val_acc: 0.8000
Epoch 2/10
 928/1000 [==========================>...] - ETA: 0s - loss: 0.4208 - acc:
0.8761
Epoch 00002: saving model to training_1/cp.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A09A90EB8>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
1000/1000 [==============================] - 1s 518us/sample - loss: 0.4182 -
acc: 0.8780 - val_loss: 0.5596 - val_acc: 0.8210
Epoch 3/10
 992/1000 [============================>.] - ETA: 0s - loss: 0.2737 - acc:
0.9345
Epoch 00003: saving model to training_1/cp.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A09A90EB8>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
1000/1000 [==============================] - 1s 652us/sample - loss: 0.2808 -
acc: 0.9330 - val_loss: 0.4468 - val_acc: 0.8590
```

```
Epoch 4/10
 896/1000 [=========================>...] - ETA: 0s - loss: 0.2007 - acc:
0.9565
Epoch 00004: saving model to training_1/cp.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A09A90EB8>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
1000/1000 [==============================] - 1s 608us/sample - loss: 0.2093 -
acc: 0.9530 - val_loss: 0.4372 - val_acc: 0.8610
Epoch 5/10
 864/1000 [========================>...] - ETA: 0s - loss: 0.1527 - acc:
0.9641
Epoch 00005: saving model to training_1/cp.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A09A90EB8>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
1000/1000 [==============================] - 1s 542us/sample - loss: 0.1564 -
acc: 0.9610 - val_loss: 0.4062 - val_acc: 0.8690
Epoch 6/10
 960/1000 [===========================>..] - ETA: 0s - loss: 0.1175 - acc:
0.9760
Epoch 00006: saving model to training_1/cp.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A09A90EB8>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
1000/1000 [==============================] - 1s 519us/sample - loss: 0.1171 -
acc: 0.9770 - val_loss: 0.4159 - val_acc: 0.8680
Epoch 7/10
 960/1000 [===========================>..] - ETA: 0s - loss: 0.0806 - acc:
0.9917
Epoch 00007: saving model to training_1/cp.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A09A90EB8>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.
```

```
Consider using a TensorFlow optimizer from `tf.train`.
1000/1000 [==============================] - 1s 519us/sample - loss: 0.0807 -
acc: 0.9920 - val_loss: 0.4018 - val_acc: 0.8660
Epoch 8/10
 960/1000 [===========================>..] - ETA: 0s - loss: 0.0624 - acc:
0.9917
Epoch 00008: saving model to training_1/cp.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A09A90EB8>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
1000/1000 [==============================] - 1s 541us/sample - loss: 0.0615 -
acc: 0.9920 - val_loss: 0.4067 - val_acc: 0.8690
Epoch 9/10
 960/1000 [===========================>..] - ETA: 0s - loss: 0.0470 - acc:
0.9979
Epoch 00009: saving model to training_1/cp.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A09A90EB8>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
1000/1000 [==============================] - 1s 518us/sample - loss: 0.0493 -
acc: 0.9970 - val_loss: 0.4129 - val_acc: 0.8700
Epoch 10/10
 864/1000 [========================>...] - ETA: 0s - loss: 0.0380 - acc:
0.9988
Epoch 00010: saving model to training_1/cp.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A09A90EB8>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
1000/1000 [==============================] - 1s 576us/sample - loss: 0.0381 -
acc: 0.9990 - val_loss: 0.3890 - val_acc: 0.8800
```

[18]:
```
<tensorflow.python.keras.callbacks.History at 0x19a09fc6c50>
```

[5]:
```
model = create_model()
```

```
loss, acc = model.evaluate(test_images, test_labels)
print("Untrained model, accuracy: {:5.2f}%".format(100*acc))
```

```
1000/1000 [==============================] - 0s 88us/sample - loss: 2.3514 -
acc: 0.0650
Untrained model, accuracy:  6.50%
```

By loading from the checkpoint, we can see it is indeed the same model trained before.

[6]:
```
model.load_weights(checkpoint_path)

loss,acc = model.evaluate(test_images, test_labels)
print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

```
1000/1000 [==============================] - 0s 45us/sample - loss: 0.4100 -
acc: 0.8650
Restored model, accuracy: 86.50%
```

[7]:
```
# include the epoch in the file name. (uses `str.format`)
checkpoint_path = "training_2/cp-{epoch:04d}.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

cp_callback = tf.keras.callbacks.ModelCheckpoint(
    checkpoint_path, verbose=1, save_weights_only=True,
    # Save weights, every 5-epochs.
    period=5)

model = create_model()
model.save_weights(checkpoint_path.format(epoch=0))
model.fit(train_images, train_labels,
          epochs = 50, callbacks = [cp_callback],
          validation_data = (test_images,test_labels),
          verbose=0)
```

```
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7E70FCC0>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.

Epoch 00005: saving model to training_2/cp-0005.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7E70FCC0>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
```

```
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.

Epoch 00010: saving model to training_2/cp-0010.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7E70FCC0>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.

Epoch 00015: saving model to training_2/cp-0015.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7E70FCC0>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.

Epoch 00020: saving model to training_2/cp-0020.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7E70FCC0>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.

Epoch 00025: saving model to training_2/cp-0025.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7E70FCC0>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.

Epoch 00030: saving model to training_2/cp-0030.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7E70FCC0>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
```

```
Epoch 00035: saving model to training_2/cp-0035.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7E70FCC0>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.

Epoch 00040: saving model to training_2/cp-0040.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7E70FCC0>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.

Epoch 00045: saving model to training_2/cp-0045.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7E70FCC0>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.

Epoch 00050: saving model to training_2/cp-0050.ckpt
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7E70FCC0>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
```

[7]:
```
<tensorflow.python.keras.callbacks.History at 0x19a7f216be0>
```

[8]:
```
! dir {checkpoint_dir}
```

```
 Volume in drive D is DATA
 Volume Serial Number is 881A-0976

 Directory of
D:\Projetos\DeepLearning_MiniProjects\MNIST_SaveRestoreModels\training_2

12/06/2019  18:02    <DIR>          .
```

```
12/06/2019  18:02    <DIR>          ..
12/06/2019  18:02                81 checkpoint
12/06/2019  18:02         1ß631ß508 cp-0000.ckpt.data-00000-of-00001
12/06/2019  18:02               648 cp-0000.ckpt.index
12/06/2019  18:02         1ß631ß508 cp-0005.ckpt.data-00000-of-00001
12/06/2019  18:02               648 cp-0005.ckpt.index
12/06/2019  18:02         1ß631ß508 cp-0010.ckpt.data-00000-of-00001
12/06/2019  18:02               648 cp-0010.ckpt.index
12/06/2019  18:02         1ß631ß508 cp-0015.ckpt.data-00000-of-00001
12/06/2019  18:02               648 cp-0015.ckpt.index
12/06/2019  18:02         1ß631ß508 cp-0020.ckpt.data-00000-of-00001
12/06/2019  18:02               648 cp-0020.ckpt.index
12/06/2019  18:02         1ß631ß508 cp-0025.ckpt.data-00000-of-00001
12/06/2019  18:02               648 cp-0025.ckpt.index
12/06/2019  18:02         1ß631ß508 cp-0030.ckpt.data-00000-of-00001
12/06/2019  18:02               648 cp-0030.ckpt.index
12/06/2019  18:02         1ß631ß508 cp-0035.ckpt.data-00000-of-00001
12/06/2019  18:02               648 cp-0035.ckpt.index
12/06/2019  18:02         1ß631ß508 cp-0040.ckpt.data-00000-of-00001
12/06/2019  18:02               648 cp-0040.ckpt.index
12/06/2019  18:02         1ß631ß508 cp-0045.ckpt.data-00000-of-00001
12/06/2019  18:02               648 cp-0045.ckpt.index
12/06/2019  18:02         1ß631ß508 cp-0050.ckpt.data-00000-of-00001
12/06/2019  18:02               648 cp-0050.ckpt.index
              23 File(s)     17ß953ß797 bytes
               2 Dir(s)  667ß718ß631ß424 bytes free
```

[9]:
```
latest = tf.train.latest_checkpoint(checkpoint_dir)
latest
```

[9]:
```
'training_2\\cp-0050.ckpt'
```

[10]:
```
model = create_model()
model.load_weights(latest)
loss, acc = model.evaluate(test_images, test_labels)
print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

```
1000/1000 [==============================] - 0s 97us/sample - loss: 0.4760 -
acc: 0.8730
Restored model, accuracy: 87.30%
```

1.2 Manually save weigths

[11]:
```
# Save the weights
model.save_weights('./checkpoints/my_checkpoint')

# Restore the weights
model = create_model()
```

```
model.load_weights('./checkpoints/my_checkpoint')

loss,acc = model.evaluate(test_images, test_labels)
print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

```
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A7F25F3C8>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
1000/1000 [==============================] - 0s 106us/sample - loss: 0.4760 -
acc: 0.8730
Restored model, accuracy: 87.30%
```

1.3 Save the entire model

The entire model can be saved to a file that contains: * the weight values * the model's configuration * the optimizer's configuration (depends on set up).

This allows you to checkpoint a model and resume training later—from the exact same state—without access to the original code.

Saving a fully-functional model is very useful—you can load them in TensorFlow.js (HDF5, Saved Model) and then train and run them in web browsers, or convert them to run on mobile devices using TensorFlow Lite (HDF5, Saved Model).

HDF5 file

[12]:
```
model = create_model()

model.fit(train_images, train_labels, epochs=5)

# Save entire model to a HDF5 file
model.save('my_model.h5')
```

```
Epoch 1/5
1000/1000 [==============================] - 0s 451us/sample - loss: 1.0890 -
acc: 0.6980
Epoch 2/5
1000/1000 [==============================] - 0s 305us/sample - loss: 0.4127 -
acc: 0.8890
Epoch 3/5
1000/1000 [==============================] - 0s 319us/sample - loss: 0.2822 -
acc: 0.9240
Epoch 4/5
1000/1000 [==============================] - 0s 308us/sample - loss: 0.2002 -
acc: 0.9500
Epoch 5/5
```

```
1000/1000 [==============================] - 0s 294us/sample - loss: 0.1470 -
acc: 0.9720
```

[13]:
```python
# Recreate the exact same model, including weights and optimizer.
new_model = keras.models.load_model('my_model.h5')
new_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_12 (Dense)             (None, 512)               401920
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_13 (Dense)             (None, 10)                5130
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

[14]:
```python
loss, acc = new_model.evaluate(test_images, test_labels)
print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

```
1000/1000 [==============================] - 0s 119us/sample - loss: 0.4141 -
acc: 0.8730
Restored model, accuracy: 87.30%
```

Currently, it is not able to save TensorFlow optimizers (from tf.train).

saved_model

[22]:
```python
model = create_model()

model.fit(train_images, train_labels, epochs=5)

loss, acc = model.evaluate(test_images, test_labels)
print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

```
Epoch 1/5
1000/1000 [==============================] - 1s 659us/sample - loss: 1.1400 -
acc: 0.6720
Epoch 2/5
1000/1000 [==============================] - 0s 486us/sample - loss: 0.4206 -
acc: 0.8790
Epoch 3/5
1000/1000 [==============================] - 0s 434us/sample - loss: 0.2691 -
```

```
acc: 0.9310
Epoch 4/5
1000/1000 [==============================] - 0s 429us/sample - loss: 0.1963 -
acc: 0.9590
Epoch 5/5
1000/1000 [==============================] - 0s 495us/sample - loss: 0.1475 -
acc: 0.9660
1000/1000 [==============================] - 0s 189us/sample - loss: 0.4103 -
acc: 0.8660
Restored model, accuracy: 86.60%
```

[23]:
```
saved_model_path = tf.contrib.saved_model.save_keras_model(model, "./
 ↪saved_models")
```

```
WARNING:tensorflow:This model was compiled with a Keras optimizer
(<tensorflow.python.keras.optimizers.Adam object at 0x0000019A0B8B7A58>) but is
being saved in TensorFlow format with `save_weights`. The model's weights will
be saved, but unlike with TensorFlow optimizers in the TensorFlow format the
optimizer's state will not be saved.

Consider using a TensorFlow optimizer from `tf.train`.
WARNING:tensorflow:Model was compiled with an optimizer, but the optimizer is
not from `tf.train` (e.g. `tf.train.AdagradOptimizer`). Only the serving graph
was exported. The train and evaluate graphs were not added to the SavedModel.
INFO:tensorflow:Signatures INCLUDED in export for Classify: None
INFO:tensorflow:Signatures INCLUDED in export for Regress: None
INFO:tensorflow:Signatures INCLUDED in export for Predict: ['serving_default']
INFO:tensorflow:Signatures INCLUDED in export for Train: None
INFO:tensorflow:Signatures INCLUDED in export for Eval: None
INFO:tensorflow:No assets to save.
INFO:tensorflow:No assets to write.
INFO:tensorflow:SavedModel written to: ./saved_models\1560361247\saved_model.pb
```

[24]:
```
new_model = tf.contrib.saved_model.load_keras_model(saved_model_path)
new_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_20 (Dense)             (None, 512)               401920
_________________________________________________________________
dropout_10 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_21 (Dense)             (None, 10)                5130
=================================================================
Total params: 407,050
Trainable params: 407,050
```

```
Non-trainable params: 0
_________________________________________________________________
```

[25]:
```
# The model has to be compiled before evaluating.
# This step is not required if the saved model is only being deployed.

new_model.compile(optimizer=tf.keras.optimizers.Adam(),
              loss=tf.keras.losses.sparse_categorical_crossentropy,
              metrics=['accuracy'])

# Evaluate the restored model.
loss, acc = new_model.evaluate(test_images, test_labels)
print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

```
1000/1000 [==============================] - 0s 202us/sample - loss: 0.4103 -
acc: 0.8660
Restored model, accuracy: 86.60%
```