Exercise11

May 30, 2022

0.1 Exercise-11

0.2 Index No - 190621M

0.3 Name - K. Thanushan

0.3.1 Question 1.

[ ]:
```
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
```

[ ]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models
import numpy as np
import matplotlib.pyplot as plt

mnist = keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Padding
paddings = tf.constant([[0, 0], [2, 2], [2, 2]])
train_images = tf.pad(train_images, paddings, constant_values=0)
test_images = tf.pad(test_images, paddings, constant_values=0)

print('train_images.shape: ', train_images.shape)
print('train_labels.shape: ', train_labels.shape)
print('test_images.shape:', test_images.shape)
print('test_labels.shape:', test_labels.shape)
class_names = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

train_images = tf.dtypes.cast(train_images, tf.float32)
test_images = tf.dtypes.cast(test_images, tf.float32)
train_images, test_images = train_images[..., np.newaxis]/255.0, test_images[...
 ↪, np.newaxis]/255.0
```

```
train_images.shape:  (60000, 32, 32)
train_labels.shape:  (60000,)
test_images.shape: (10000, 32, 32)
test_labels.shape: (10000,)
```

[ ]:
```
model = models.Sequential()
model.add(layers.Conv2D(6, (5,5), activation='relu', input_shape=(32, 32, 1)))
model.add(layers.AveragePooling2D((2,2)))
model.add(layers.Conv2D(16, (5,5), activation='relu'))
model.add(layers.AveragePooling2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(120, activation='relu'))
model.add(layers.Dense(84, activation='relu'))
model.add(layers.Dense(10))

model.compile(optimizer = 'adam', loss = tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits = True), metrics = ['accuracy'])
print(model.summary())

model.fit(train_images, train_labels, epochs=5)
test_loss, test_accuracy = model.evaluate(test_images, test_labels, verbose = 2)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 6)         156

 average_pooling2d (AverageP  (None, 14, 14, 6)        0
 ooling2D)

 conv2d_1 (Conv2D)           (None, 10, 10, 16)        2416

 average_pooling2d_1 (Averag  (None, 5, 5, 16)         0
 ePooling2D)

 flatten (Flatten)           (None, 400)               0

 dense (Dense)               (None, 120)               48120

 dense_1 (Dense)             (None, 84)                10164

 dense_2 (Dense)             (None, 10)                850

=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/5
1875/1875 [==============================] - 12s 6ms/step - loss: 0.2146 -
accuracy: 0.9347
```

```
Epoch 2/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0684 -
accuracy: 0.9786
Epoch 3/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0482 -
accuracy: 0.9847
Epoch 4/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0375 -
accuracy: 0.9880
Epoch 5/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0316 -
accuracy: 0.9898
313/313 - 1s - loss: 0.0454 - accuracy: 0.9857 - 678ms/epoch - 2ms/step
```

0.3.2 Question 2.

[ ]:
```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import cifar10, mnist
import tensorflow as tf
import matplotlib.pyplot as plt
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.
 ↪load_data()

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog',␣
 ↪'horse', 'ship', 'truck']
```

[ ]:
```
model = models.Sequential()
model.add(layers.Conv2D(32,(5,5),activation = 'relu',input_shape = (32,32,3)))
model.add(layers.MaxPool2D((2,2)))
model.add(layers.Conv2D(64,(3,3),activation = 'relu'))
model.add(layers.MaxPool2D((2,2)))
model.add(layers.Conv2D(128,(3,3),activation = 'relu'))
model.add(layers.MaxPool2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(64,activation = 'relu'))
model.add(layers.Dense(10))


model.compile(optimizer = 'adam',loss = tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),metrics = ['accuracy'])
print(model.summary)
model.fit(train_images,train_labels,epochs = 5)
test_loss, test_accuracy = model.evaluate(test_images,test_labels,verbose = 2)
```

```
<bound method Model.summary of <keras.engine.sequential.Sequential object at
0x0000021F0B52A5F0>>
Epoch 1/5
1563/1563 [==============================] - 34s 22ms/step - loss: 1.5643 -
accuracy: 0.4226
Epoch 2/5
1563/1563 [==============================] - 34s 22ms/step - loss: 1.2094 -
accuracy: 0.5691
Epoch 3/5
1563/1563 [==============================] - 35s 22ms/step - loss: 1.0441 -
accuracy: 0.6330
Epoch 4/5
1563/1563 [==============================] - 36s 23ms/step - loss: 0.9379 -
accuracy: 0.6734
Epoch 5/5
1563/1563 [==============================] - 34s 22ms/step - loss: 0.8553 -
accuracy: 0.7004
313/313 - 2s - loss: 0.9266 - accuracy: 0.6764 - 2s/epoch - 7ms/step
```

0.3.3 Question 3.

[ ]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models
import numpy as np
import matplotlib.pyplot as plt

mnist = keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Padding
paddings = tf.constant([[0, 0], [2, 2], [2, 2]])
train_images = tf.pad(train_images, paddings, constant_values=0)
test_images = tf.pad(test_images, paddings, constant_values=0)

print('train_images.shape: ', train_images.shape)
print('train_labels.shape: ', train_labels.shape)
print('test_images.shape:', test_images.shape)
print('test_labels.shape:', test_labels.shape)
class_names = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

train_images = tf.dtypes.cast(train_images, tf.float32)
test_images = tf.dtypes.cast(test_images, tf.float32)
train_images, test_images = train_images[..., np.newaxis]/255.0, test_images[...
 ↪, np.newaxis]/255.0
```

```
train_images.shape:  (60000, 32, 32)
train_labels.shape:  (60000,)
```

```
test_images.shape: (10000, 32, 32)
test_labels.shape: (10000,)
```

[ ]:
```
model_base = models.Sequential()
model_base.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(32, 32,␣
 ↪1)))
model_base.add(layers.MaxPool2D((2,2)))
model_base.add(layers.Conv2D(64, (3,3), activation='relu'))
model_base.add(layers.MaxPool2D((2,2)))
model_base.add(layers.Flatten())
model_base.add(layers.Dense(64, activation='relu'))
model_base.add(layers.Dense(10))

model_base.compile(optimizer = 'adam', loss = tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits = True), metrics = ['accuracy'])
print(model_base.summary())

model_base.fit(train_images, train_labels, epochs=2)
test_loss, test_accuracy = model_base.evaluate(test_images, test_labels,␣
 ↪verbose = 2)
model_base.save_weights('saved_weights/')
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 30, 30, 32)        320

 max_pooling2d_6 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_9 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_7 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_3 (Flatten)         (None, 2304)              0

 dense_7 (Dense)             (None, 64)                147520

 dense_8 (Dense)             (None, 10)                650

=================================================================
Total params: 166,986
Trainable params: 166,986
Non-trainable params: 0
_________________________________________________________________
None
```

```
Epoch 1/2
1875/1875 [==============================] - 30s 16ms/step - loss: 0.1273 -
accuracy: 0.9612
Epoch 2/2
1875/1875 [==============================] - 30s 16ms/step - loss: 0.0424 -
accuracy: 0.9868
313/313 - 1s - loss: 0.0464 - accuracy: 0.9851 - 1s/epoch - 4ms/step
```

0.3.4 Question 4.

[ ]:
```
model_lw = models.Sequential()
model_lw.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(32, 32,␣
↪1)))
model_lw.add(layers.MaxPool2D((2,2)))
model_lw.add(layers.Conv2D(64, (3,3), activation='relu'))
model_lw.add(layers.MaxPool2D((2,2)))
model_lw.add(layers.Flatten())
model_lw.add(layers.Dense(64, activation='relu'))
model_lw.add(layers.Dense(10))

model_lw.compile(optimizer = 'adam', loss = tf.keras.losses.
↪SparseCategoricalCrossentropy(from_logits = True), metrics = ['accuracy'])
print(model_lw.summary())

model_lw.load_weights('saved_weights/')
model_lw.fit(train_images, train_labels, epochs=2)
test_loss, test_accuracy = model_lw.evaluate(test_images, test_labels, verbose␣
↪= 2)
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 30, 30, 32)        320

 max_pooling2d_8 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_11 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_9 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_4 (Flatten)         (None, 2304)              0

 dense_9 (Dense)             (None, 64)                147520

 dense_10 (Dense)            (None, 10)                650
```

```
=================================================================
Total params: 166,986
Trainable params: 166,986
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/2
1875/1875 [==============================] - 33s 17ms/step - loss: 0.0292 -
accuracy: 0.9908
Epoch 2/2
1875/1875 [==============================] - 31s 17ms/step - loss: 0.0212 -
accuracy: 0.9930
313/313 - 1s - loss: 0.0319 - accuracy: 0.9906 - 1s/epoch - 4ms/step
```

[ ]:
```
model_lw.save('saved_model/')
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op while saving (showing 2 of 2). These functions will
not be directly callable after loading.

INFO:tensorflow:Assets written to: saved_model/assets

INFO:tensorflow:Assets written to: saved_model/assets
```

0.3.5 Question 5.

[ ]:
```
#Loading the model
model_ld = keras.models.load_model('saved_model/')
print(model_ld.summary())
model_ld.evaluate(test_images, test_labels, verbose = 2)
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10 (Conv2D)          (None, 30, 30, 32)        320

 max_pooling2d_8 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_11 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_9 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_4 (Flatten)         (None, 2304)              0

 dense_9 (Dense)             (None, 64)                147520
```

```
 dense_10 (Dense)            (None, 10)                650

=================================================================
Total params: 166,986
Trainable params: 166,986
Non-trainable params: 0
_________________________________________________________________
None
313/313 - 1s - loss: 0.0319 - accuracy: 0.9906 - 1s/epoch - 4ms/step
```

[ ]:
```
[0.03192006051540375, 0.9905999898910522]
```

0.3.6 Question 6. - Fine Tuning

[ ]:
```
base_inputs = model_ld.layers[0].input
base_outputs = model_ld.layers[-2].output
output = layers.Dense(10)(base_outputs)

new_model = keras.Model(inputs = base_inputs, outputs = output)
new_model.compile(optimizer = keras.optimizers.Adam(), loss = tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits = True), metrics = ['accuracy'])
print(new_model.summary())

new_model.load_weights('saved_weights/')
new_model.fit(train_images, train_labels, epochs=3)
test_loss, test_accuracy = new_model.evaluate(test_images, test_labels, verbose␣
 ↪= 2)
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_10_input (InputLayer  [(None, 32, 32, 1)]      0
 )

 conv2d_10 (Conv2D)          (None, 30, 30, 32)        320

 max_pooling2d_8 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_11 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_9 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_4 (Flatten)         (None, 2304)              0

 dense_9 (Dense)             (None, 64)                147520
```

```
 dense_11 (Dense)            (None, 10)                650

=================================================================
Total params: 166,986
Trainable params: 166,986
Non-trainable params: 0

_________________________________________________________________
None
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this
object graph. For example, in the saved checkpoint object, `model.layer.weight`
and `model.layer_copy.weight` reference the same variable, while in the current
object these are two different variables. The referenced variables
are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x0000021F426EE2F0> and
<keras.engine.input_layer.InputLayer object at 0x0000021F43CE4EB0>).

WARNING:tensorflow:Inconsistent references when loading the checkpoint into this
object graph. For example, in the saved checkpoint object, `model.layer.weight`
and `model.layer_copy.weight` reference the same variable, while in the current
object these are two different variables. The referenced variables
are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x0000021F426EE2F0> and
<keras.engine.input_layer.InputLayer object at 0x0000021F43CE4EB0>).

WARNING:tensorflow:Inconsistent references when loading the checkpoint into this
object graph. For example, in the saved checkpoint object, `model.layer.weight`
and `model.layer_copy.weight` reference the same variable, while in the current
object these are two different variables. The referenced variables
are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x0000021F426EDBA0> and
<keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x0000021F426EF2E0>).

WARNING:tensorflow:Inconsistent references when loading the checkpoint into this
object graph. For example, in the saved checkpoint object, `model.layer.weight`
and `model.layer_copy.weight` reference the same variable, while in the current
object these are two different variables. The referenced variables
are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x0000021F426EDBA0> and
<keras.layers.pooling.max_pooling2d.MaxPooling2D object at 0x0000021F426EF2E0>).

WARNING:tensorflow:Inconsistent references when loading the checkpoint into this
object graph. For example, in the saved checkpoint object, `model.layer.weight`
and `model.layer_copy.weight` reference the same variable, while in the current
object these are two different variables. The referenced variables
are:(<keras.layers.core.dense.Dense object at 0x0000021F41162F80> and
<keras.layers.reshaping.flatten.Flatten object at 0x0000021F426EE050>).

WARNING:tensorflow:Inconsistent references when loading the checkpoint into this
object graph. For example, in the saved checkpoint object, `model.layer.weight`
and `model.layer_copy.weight` reference the same variable, while in the current
object these are two different variables. The referenced variables
are:(<keras.layers.core.dense.Dense object at 0x0000021F41162F80> and
<keras.layers.reshaping.flatten.Flatten object at 0x0000021F426EE050>).
```

```
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this
object graph. For example, in the saved checkpoint object, `model.layer.weight`
and `model.layer_copy.weight` reference the same variable, while in the current
object these are two different variables. The referenced variables
are:(<keras.layers.core.dense.Dense object at 0x0000021F438615A0> and
<keras.layers.core.dense.Dense object at 0x0000021F41162F80>).

WARNING:tensorflow:Inconsistent references when loading the checkpoint into this
object graph. For example, in the saved checkpoint object, `model.layer.weight`
and `model.layer_copy.weight` reference the same variable, while in the current
object these are two different variables. The referenced variables
are:(<keras.layers.core.dense.Dense object at 0x0000021F438615A0> and
<keras.layers.core.dense.Dense object at 0x0000021F41162F80>).

Epoch 1/3
1875/1875 [==============================] - 30s 16ms/step - loss: 0.0281 -
accuracy: 0.9912
Epoch 2/3
1875/1875 [==============================] - 29s 15ms/step - loss: 0.0215 -
accuracy: 0.9934
Epoch 3/3
1875/1875 [==============================] - 29s 15ms/step - loss: 0.0157 -
accuracy: 0.9947
313/313 - 1s - loss: 0.0341 - accuracy: 0.9905 - 1s/epoch - 4ms/step
```

0.3.7 Question 7.

[ ]:
```
#Transfer Learning
model_for_tl = keras.models.load_model('saved_model/')
model_for_tl.trainable = False
for layer in model_for_tl.layers:
    assert layer.trainable == False

base_inputs = model_for_tl.layers[0].input
base_outputs = model_for_tl.layers[-2].output
output = layers.Dense(10)(base_outputs)

new_model = keras.Model(inputs = base_inputs, outputs = output)
new_model.compile(optimizer = keras.optimizers.Adam(), loss = tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits = True), metrics = ['accuracy'])

new_model.fit(train_images, train_labels, epochs=3)
test_loss, test_accuracy = new_model.evaluate(test_images, test_labels, verbose␣
 ↪= 2)
```

```
Epoch 1/3
1875/1875 [==============================] - 9s 4ms/step - loss: 0.2250 -
accuracy: 0.9472
Epoch 2/3
```

```
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0162 -
accuracy: 0.9963
Epoch 3/3
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0109 -
accuracy: 0.9972
313/313 - 1s - loss: 0.0261 - accuracy: 0.9912 - 1s/epoch - 4ms/step
```

0.3.8 Question 8.

[ ]:
```
x = tf.random.normal(shape = (5, 224, 224, 3))
y = tf.constant([0, 1, 2, 3, 4])
rn_model = keras.applications.resnet_v2.ResNet50V2(include_top=True)
rn_model.trainable = False
for layer in rn_model.layers:
    assert layer.trainable == False

base_inputs = rn_model.layers[0].input
base_outputs = rn_model.layers[-2].output
output = layers.Dense(5)(base_outputs)

new_model = keras.Model(inputs = base_inputs, outputs = output)
new_model.compile(optimizer = keras.optimizers.Adam(), loss = tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits = True), metrics = ['accuracy'])

new_model.fit(x, y, epochs=15, verbose=2)
```

```
Epoch 1/15
1/1 - 3s - loss: 1.6548 - accuracy: 0.2000 - 3s/epoch - 3s/step
Epoch 2/15
1/1 - 0s - loss: 1.5944 - accuracy: 0.2000 - 301ms/epoch - 301ms/step
Epoch 3/15
1/1 - 0s - loss: 1.5586 - accuracy: 0.2000 - 264ms/epoch - 264ms/step
Epoch 4/15
1/1 - 0s - loss: 1.5330 - accuracy: 0.6000 - 268ms/epoch - 268ms/step
Epoch 5/15
1/1 - 0s - loss: 1.5072 - accuracy: 0.6000 - 270ms/epoch - 270ms/step
Epoch 6/15
1/1 - 0s - loss: 1.4783 - accuracy: 0.6000 - 318ms/epoch - 318ms/step
Epoch 7/15
1/1 - 0s - loss: 1.4466 - accuracy: 0.8000 - 280ms/epoch - 280ms/step
Epoch 8/15
1/1 - 0s - loss: 1.4134 - accuracy: 0.8000 - 276ms/epoch - 276ms/step
Epoch 9/15
1/1 - 0s - loss: 1.3802 - accuracy: 0.8000 - 269ms/epoch - 269ms/step
Epoch 10/15
1/1 - 0s - loss: 1.3483 - accuracy: 1.0000 - 310ms/epoch - 310ms/step
Epoch 11/15
1/1 - 0s - loss: 1.3181 - accuracy: 1.0000 - 301ms/epoch - 301ms/step
```

```
Epoch 12/15
1/1 - 0s - loss: 1.2895 - accuracy: 1.0000 - 288ms/epoch - 288ms/step
Epoch 13/15
1/1 - 0s - loss: 1.2617 - accuracy: 1.0000 - 276ms/epoch - 276ms/step
Epoch 14/15
1/1 - 0s - loss: 1.2337 - accuracy: 1.0000 - 334ms/epoch - 334ms/step
Epoch 15/15
1/1 - 0s - loss: 1.2053 - accuracy: 1.0000 - 272ms/epoch - 272ms/step
```

[ ]: <keras.callbacks.History at 0x21f01ff19f0>