Index number : 190026T

Name : AHAMED M.I.I

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

In [ ]:
```
#1)
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,)
```

In [ ]:
```
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_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 28, 28, 6)         156

 average_pooling2d_2 (Averag  (None, 14, 14, 6)        0
 ePooling2D)

 conv2d_3 (Conv2D)           (None, 10, 10, 16)        2416

 average_pooling2d_3 (Averag  (None, 5, 5, 16)         0
 ePooling2D)

 flatten_1 (Flatten)         (None, 400)               0

 dense_3 (Dense)             (None, 120)               48120

 dense_4 (Dense)             (None, 84)                10164

 dense_5 (Dense)             (None, 10)                850

=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/5
1875/1875 [==============================] - 17s 9ms/step - loss: 0.2109 - accuracy: 0.9355
Epoch 2/5
1875/1875 [==============================] - 16s 8ms/step - loss: 0.0682 - accuracy: 0.9789
Epoch 3/5
1875/1875 [==============================] - 16s 8ms/step - loss: 0.0479 - accuracy: 0.9847
Epoch 4/5
1875/1875 [==============================] - 16s 9ms/step - loss: 0.0362 - accuracy: 0.9880
Epoch 5/5
1875/1875 [==============================] - 17s 9ms/step - loss: 0.0304 - accuracy: 0.9904
313/313 - 1s - loss: 0.0361 - accuracy: 0.9889 - 1s/epoch - 4ms/step
```

In [ ]:
```
#2)

from tensorflow.keras.datasets import cifar10, mnist
```

```
(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']
```

In [ ]:

```
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 0x000001D6B5D60730>>
Epoch 1/5
1563/1563 [==============================] - 37s 23ms/step - loss: 1.5605 - accuracy: 0.4320
Epoch 2/5
1563/1563 [==============================] - 33s 21ms/step - loss: 1.1765 - accuracy: 0.5845
Epoch 3/5
1563/1563 [==============================] - 35s 22ms/step - loss: 1.0099 - accuracy: 0.6495
Epoch 4/5
1563/1563 [==============================] - 35s 22ms/step - loss: 0.9060 - accuracy: 0.6849
Epoch 5/5
1563/1563 [==============================] - 34s 22ms/step - loss: 0.8197 - accuracy: 0.7160
313/313 - 2s - loss: 0.9179 - accuracy: 0.6830 - 2s/epoch - 7ms/step
```

In [ ]:

```
#3)

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,)
```

In [ ]:

```
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_7 (Conv2D)           (None, 30, 30, 32)        320

 max_pooling2d_3 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_8 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_4 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_3 (Flatten)         (None, 2304)              0

 dense_8 (Dense)             (None, 64)                147520

 dense_9 (Dense)             (None, 10)                650

=================================================================
Total params: 166,986
```

```
Trainable params: 166,986
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/2
1875/1875 [==============================] - 32s 17ms/step - loss: 0.1425 - accuracy: 0.9558
Epoch 2/2
1875/1875 [==============================] - 32s 17ms/step - loss: 0.0457 - accuracy: 0.9859
313/313 - 2s - loss: 0.0398 - accuracy: 0.9874 - 2s/epoch - 5ms/step
```

In [ ]:

```
#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_9 (Conv2D)           (None, 30, 30, 32)        320

 max_pooling2d_5 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_10 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_6 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_4 (Flatten)         (None, 2304)              0

 dense_10 (Dense)            (None, 64)                147520

 dense_11 (Dense)            (None, 10)                650

=================================================================
Total params: 166,986
Trainable params: 166,986
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/2
1875/1875 [==============================] - 32s 17ms/step - loss: 0.0308 - accuracy: 0.9902
Epoch 2/2
1875/1875 [==============================] - 32s 17ms/step - loss: 0.0227 - accuracy: 0.9925
313/313 - 2s - loss: 0.0394 - accuracy: 0.9867 - 2s/epoch - 5ms/step
```

In [ ]:

```
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). Th
ese functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: saved_model/assets
INFO:tensorflow:Assets written to: saved_model/assets
```

In [ ]:

```
#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_9 (Conv2D)           (None, 30, 30, 32)        320

 max_pooling2d_5 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_10 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_6 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_4 (Flatten)         (None, 2304)              0

 dense_10 (Dense)            (None, 64)                147520

 dense_11 (Dense)            (None, 10)                650

=================================================================
Total params: 166,986
Trainable params: 166,986
Non-trainable params: 0
_________________________________________________________________
None
313/313 - 2s - loss: 0.0394 - accuracy: 0.9867 - 2s/epoch - 6ms/step
```

Out[ ]: [0.03938652202486992, 0.9866999983787537]

In [ ]:
```
#6)

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 =
print(model_lw.summary())

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

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 30, 30, 32)        320

 max_pooling2d_5 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_10 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_6 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 flatten_4 (Flatten)         (None, 2304)              0

 dense_10 (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 objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x000001D6B61C30A0> and <keras.engine.input_laye
r.InputLayer object at 0x000001D6B630FDC0>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x000001D6B61C30A0> and <keras.engine.input_laye
r.InputLayer object at 0x000001D6B630FDC0>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x000001D6B617F6A0> and <keras.layers.pooling.ma
x_pooling2d.MaxPooling2D object at 0x000001D6B3875670>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x000001D6B617F6A0> and <keras.layers.pooling.ma
x_pooling2d.MaxPooling2D object at 0x000001D6B3875670>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.core.dense.Dense object at 0x000001D6B612B850> and <keras.layers.reshaping.flatten.Fla
tten object at 0x000001D6B612BE50>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.core.dense.Dense object at 0x000001D6B612B850> and <keras.layers.reshaping.flatten.Fla
tten object at 0x000001D6B612BE50>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.core.dense.Dense object at 0x000001D6B8453640> and <keras.layers.core.dense.Dense obje
ct at 0x000001D6B612B850>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.core.dense.Dense object at 0x000001D6B8453640> and <keras.layers.core.dense.Dense obje
ct at 0x000001D6B612B850>).
Epoch 1/2
1875/1875 [==============================] - 31s 16ms/step - loss: 0.0318 - accuracy: 0.9900
Epoch 2/2
1875/1875 [==============================] - 31s 17ms/step - loss: 0.0222 - accuracy: 0.9931
313/313 - 2s - loss: 0.0292 - accuracy: 0.9910 - 2s/epoch - 6ms/step
```

In [ ]:
```
#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 =

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

```
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
```

```
iables. The referenced variables are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x000001D6B61D9730> and <keras.engine.input_laye
r.InputLayer object at 0x000001D6B86CC4F0>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x000001D6B61D9730> and <keras.engine.input_laye
r.InputLayer object at 0x000001D6B86CC4F0>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x000001D6B61CC760> and <keras.layers.pooling.ma
x_pooling2d.MaxPooling2D object at 0x000001D6B61E3D00>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.convolutional.conv2d.Conv2D object at 0x000001D6B61CC760> and <keras.layers.pooling.ma
x_pooling2d.MaxPooling2D object at 0x000001D6B61E3D00>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.core.dense.Dense object at 0x000001D6B61BB7F0> and <keras.layers.reshaping.flatten.Fla
tten object at 0x000001D6B61BBDF0>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.core.dense.Dense object at 0x000001D6B61BB7F0> and <keras.layers.reshaping.flatten.Fla
tten object at 0x000001D6B61BBDF0>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.core.dense.Dense object at 0x000001D6B61B4190> and <keras.layers.core.dense.Dense obje
ct at 0x000001D6B61BB7F0>).
WARNING:tensorflow:Inconsistent references when loading the checkpoint into this object graph. For example, in the saved checkpoint objec
t, `model.layer.weight` and `model.layer_copy.weight` reference the same variable, while in the current object these are two different var
iables. The referenced variables are:(<keras.layers.core.dense.Dense object at 0x000001D6B61B4190> and <keras.layers.core.dense.Dense obje
ct at 0x000001D6B61BB7F0>).
Epoch 1/2
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0217 - accuracy: 0.9935
Epoch 2/2
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0197 - accuracy: 0.9938
313/313 - 2s - loss: 0.0268 - accuracy: 0.9907 - 2s/epoch - 6ms/step
```

In [ ]:

```python
#8)

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

base_inputs = resNet_model.layers[0].input
base_outputs = resNet_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)
```

```
Epoch 1/15
1/1 [==============================] - 3s 3s/step - loss: 1.9790 - accuracy: 0.2000
Epoch 2/15
1/1 [==============================] - 0s 318ms/step - loss: 1.8119 - accuracy: 0.2000
Epoch 3/15
1/1 [==============================] - 0s 304ms/step - loss: 1.6848 - accuracy: 0.2000
Epoch 4/15
1/1 [==============================] - 0s 302ms/step - loss: 1.5894 - accuracy: 0.2000
Epoch 5/15
1/1 [==============================] - 0s 304ms/step - loss: 1.5201 - accuracy: 0.2000
Epoch 6/15
1/1 [==============================] - 0s 304ms/step - loss: 1.4732 - accuracy: 0.6000
Epoch 7/15
1/1 [==============================] - 0s 305ms/step - loss: 1.4432 - accuracy: 0.4000
Epoch 8/15
1/1 [==============================] - 0s 304ms/step - loss: 1.4230 - accuracy: 0.6000
Epoch 9/15
1/1 [==============================] - 0s 309ms/step - loss: 1.4056 - accuracy: 0.6000
Epoch 10/15
1/1 [==============================] - 0s 306ms/step - loss: 1.3859 - accuracy: 0.6000
Epoch 11/15
1/1 [==============================] - 0s 302ms/step - loss: 1.3613 - accuracy: 0.6000
Epoch 12/15
1/1 [==============================] - 0s 307ms/step - loss: 1.3318 - accuracy: 0.6000
Epoch 13/15
1/1 [==============================] - 0s 303ms/step - loss: 1.2985 - accuracy: 0.8000
Epoch 14/15
1/1 [==============================] - 0s 299ms/step - loss: 1.2629 - accuracy: 0.8000
Epoch 15/15
1/1 [==============================] - 0s 303ms/step - loss: 1.2266 - accuracy: 0.8000
```

Out[ ]:

```
<keras.callbacks.History at 0x1d6c4518eb0>
```