Name : Dilshan J.V.A.P

Index number : 190144D

Exercise 11

1) Implement the LeNet5 network, as discussed in class, for MNIST.

In [ ]:
```
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 =
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_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_11 (Conv2D)          (None, 28, 28, 6)         156

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

 conv2d_12 (Conv2D)          (None, 10, 10, 16)        2416

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

 flatten_4 (Flatten)         (None, 400)               0

 dense_9 (Dense)             (None, 120)               48120

 dense_10 (Dense)            (None, 84)                10164
```

```
 dense_11 (Dense)            (None, 10)                850

=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2095 - accuracy: 0.9366
Epoch 2/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0712 - accuracy: 0.9776
Epoch 3/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0510 - accuracy: 0.9837
Epoch 4/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0390 - accuracy: 0.9881
Epoch 5/5
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0312 - accuracy: 0.9901
313/313 - 1s - loss: 0.0337 - accuracy: 0.9896 - 1s/epoch - 3ms/step
```

1. Implement a CNN for CIFAR10

In [ ]:

```
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']

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)
```

```
train_images.shape:  (50000, 32, 32, 3)
train_labels.shape:  (50000, 1)
test_images.shape: (10000, 32, 32, 3)
test_labels.shape: (10000, 1)
```

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=keras.optimizers.Adam(learning_rate = 0.001),loss = tf.keras.losses.SparseCategoricalCros
print(model.summary())

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

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_13 (Conv2D)          (None, 28, 28, 32)        2432

 max_pooling2d_8 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 conv2d_14 (Conv2D)          (None, 12, 12, 64)        18496

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

```
 conv2d_15 (Conv2D)          (None, 4, 4, 128)         73856

 max_pooling2d_10 (MaxPoolin  (None, 2, 2, 128)        0
 g2D)

 flatten_5 (Flatten)         (None, 512)               0

 dense_12 (Dense)            (None, 64)                32832

 dense_13 (Dense)            (None, 10)                650

=================================================================
Total params: 128,266
Trainable params: 128,266
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/5
1563/1563 [==============================] - 25s 16ms/step - loss: 1.5578 - accuracy: 0.4327
Epoch 2/5
1563/1563 [==============================] - 26s 17ms/step - loss: 1.1821 - accuracy: 0.5826
Epoch 3/5
1563/1563 [==============================] - 25s 16ms/step - loss: 1.0177 - accuracy: 0.6419
Epoch 4/5
1563/1563 [==============================] - 25s 16ms/step - loss: 0.9146 - accuracy: 0.6813
Epoch 5/5
1563/1563 [==============================] - 27s 17ms/step - loss: 0.8252 - accuracy: 0.7110
313/313 - 2s - loss: 0.9216 - accuracy: 0.6827 - 2s/epoch - 6ms/step
0.682699978351593
```

1. For the MINST dataset, implement the following network (call it model_base):

In [ ]:

```
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

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.Conv2D(64,(3,3),activation = 'relu'))

model_base.add(layers.Flatten())
model_base.add(layers.Dense(64,activation = 'relu'))
model_base.add(layers.Dense(10))

model_base.compile(optimizer =keras.optimizers.Adam(),loss = tf.keras.losses.SparseCategoricalCrossentropy(from_l
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/')
```

```
train_images.shape:  (60000, 32, 32)
train_labels.shape:  (60000,)
test_images.shape: (10000, 32, 32)
test_labels.shape: (10000,)
Model: "sequential_6"
_________________________________________________________________
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_16 (Conv2D)          (None, 30, 30, 32)        320

 max_pooling2d_11 (MaxPoolin  (None, 15, 15, 32)       0
 g2D)

 conv2d_17 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_12 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)

 conv2d_18 (Conv2D)          (None, 4, 4, 64)          36928

 flatten_6 (Flatten)         (None, 1024)              0

 dense_14 (Dense)            (None, 64)                65600

 dense_15 (Dense)            (None, 10)                650

=================================================================
Total params: 121,994
Trainable params: 121,994
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/2
1875/1875 [==============================] - 27s 14ms/step - loss: 0.1386 - accuracy: 0.9579
Epoch 2/2
1875/1875 [==============================] - 26s 14ms/step - loss: 0.0421 - accuracy: 0.9869
313/313 - 2s - loss: 0.0319 - accuracy: 0.9902 - 2s/epoch - 5ms/step
```

1. Create a second network with exactly the same structure as in 3. Call this model_lw. Load the weights saved in 3. Train for two epochs

In [ ]:

```
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.Conv2D(64,(3,3),activation = 'relu'))

model_lw.add(layers.Flatten())
model_lw.add(layers.Dense(64,activation = 'relu'))
model_lw.add(layers.Dense(10))

model_lw.compile(optimizer =keras.optimizers.Adam(),loss = tf.keras.losses.SparseCategoricalCrossentropy(from_log
print(model_lw.summary())

model_lw.fit(train_images,train_labels,epochs = 2)
test_loss, test_accuracy = model_lw.evaluate(test_images,test_labels,verbose = 2)
model_lw.save('saved_model/')
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_22 (Conv2D)          (None, 30, 30, 32)        320

 max_pooling2d_15 (MaxPoolin  (None, 15, 15, 32)       0
 g2D)

 conv2d_23 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_16 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)

 conv2d_24 (Conv2D)          (None, 4, 4, 64)          36928

 flatten_8 (Flatten)         (None, 1024)              0

 dense_18 (Dense)            (None, 64)                65600
```

```
 dense_19 (Dense)            (None, 10)                650

=================================================================
Total params: 121,994
Trainable params: 121,994
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/2
1875/1875 [==============================] - 25s 13ms/step - loss: 0.1303 - accuracy: 0.9599
Epoch 2/2
1875/1875 [==============================] - 25s 14ms/step - loss: 0.0429 - accuracy: 0.9865
313/313 - 2s - loss: 0.0384 - accuracy: 0.9874 - 2s/epoch - 5ms/step
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_co
mpiled_convolution_op while saving (showing 3 of 3). 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
```

1. Load this model using keras.models.load_model. Call this model_ld

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

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_22 (Conv2D)          (None, 30, 30, 32)        320

 max_pooling2d_15 (MaxPoolin  (None, 15, 15, 32)       0
 g2D)

 conv2d_23 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_16 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)

 conv2d_24 (Conv2D)          (None, 4, 4, 64)          36928

 flatten_8 (Flatten)         (None, 1024)              0

 dense_18 (Dense)            (None, 64)                65600

 dense_19 (Dense)            (None, 10)                650

=================================================================
Total params: 121,994
Trainable params: 121,994
Non-trainable params: 0
_________________________________________________________________
None
313/313 - 2s - loss: 0.0384 - accuracy: 0.9874 - 2s/epoch - 5ms/step
```

Out[ ]:
```
[0.03838663920760155, 0.9873999953269958]
```

1. Transfer learning: Load the saved model except the last layer. Connect a fresh dense layer of 10 output nodes. Train for two epochs.

In [ ]:
```
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_lo
print(new_model.summary())

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

```
Model: "model"
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_22_input (InputLayer  [(None, 32, 32, 1)]      0
 )

 conv2d_22 (Conv2D)          (None, 30, 30, 32)        320

 max_pooling2d_15 (MaxPoolin  (None, 15, 15, 32)       0
 g2D)

 conv2d_23 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_16 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)

 conv2d_24 (Conv2D)          (None, 4, 4, 64)          36928

 flatten_8 (Flatten)         (None, 1024)              0

 dense_18 (Dense)            (None, 64)                65600

 dense_20 (Dense)            (None, 10)                650

=================================================================
Total params: 121,994
Trainable params: 121,994
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/3
1875/1875 - 23s - loss: 0.0769 - accuracy: 0.9785 - 23s/epoch - 12ms/step
Epoch 2/3
1875/1875 - 23s - loss: 0.0278 - accuracy: 0.9914 - 23s/epoch - 12ms/step
Epoch 3/3
1875/1875 - 24s - loss: 0.0196 - accuracy: 0.9937 - 24s/epoch - 13ms/step
313/313 - 2s - loss: 0.0397 - accuracy: 0.9894 - 2s/epoch - 5ms/step
```

Out[ ]:
```
[0.03968558833003044, 0.9894000291824341]
```

1. Fine tuning: Load the saved model. Make the loaded layers non-trainable. Repeat the process in 6

In [ ]:
```
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_lo
print(new_model.summary())

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

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_22_input (InputLayer  [(None, 32, 32, 1)]      0
 )

 conv2d_22 (Conv2D)          (None, 30, 30, 32)        320

 max_pooling2d_15 (MaxPoolin  (None, 15, 15, 32)       0
 g2D)

 conv2d_23 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_16 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)
```

```
 conv2d_24 (Conv2D)          (None, 4, 4, 64)          36928

 flatten_8 (Flatten)         (None, 1024)              0

 dense_18 (Dense)            (None, 64)                65600

 dense_21 (Dense)            (None, 10)                650

=================================================================
Total params: 121,994
Trainable params: 650
Non-trainable params: 121,344
_________________________________________________________________
None
Epoch 1/3
1875/1875 - 8s - loss: 0.2489 - accuracy: 0.9421 - 8s/epoch - 4ms/step
Epoch 2/3
1875/1875 - 9s - loss: 0.0254 - accuracy: 0.9926 - 9s/epoch - 5ms/step
Epoch 3/3
1875/1875 - 7s - loss: 0.0204 - accuracy: 0.9941 - 7s/epoch - 4ms/step
313/313 - 2s - loss: 0.0286 - accuracy: 0.9909 - 2s/epoch - 6ms/step
```

Out[ ]:
```
[0.028552716597914696, 0.9908999800682068]
```

1. Load a pre-trained ResNet model, e.g., keras.applications.resnet_v2.ResNet50V2. Con nect an output layer of 5 nodes. Feed arrays of random numbers as input images and transfer learn.

In [ ]:
```
model_resnet = keras.applications.resnet_v2.ResNet50V2()
model_resnet.trainable=False

for layer in model_resnet.layers:
    assert layer.trainable == False

base_inputs = model_resnet.layers[0].input
base_outputs =model_resnet.layers[-2].output
output =layers.Dense(5)(base_outputs)# adding output layer of 5 nodes.

new_model = keras.Model(inputs=base_inputs,outputs=output)
new_model.compile(optimizer = keras.optimizers.Adam(),loss =tf.keras.losses.SparseCategoricalCrossentropy(from_lo

train_images = tf.random.uniform((100,224,224,3),maxval=1,dtype='float32')# random 100 images
train_labels =tf.random.uniform((100,1),maxval=4,dtype='int32')

new_model.fit(train_images,train_labels,epochs=3,verbose=2)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50v2_weights_tf_d
im_ordering_tf_kernels.h5
102869336/102869336 [==============================] - 1120s 11us/step
Epoch 1/3
4/4 - 5s - loss: 1.7879 - accuracy: 0.1000 - 5s/epoch - 1s/step
Epoch 2/3
4/4 - 3s - loss: 1.5284 - accuracy: 0.2600 - 3s/epoch - 849ms/step
Epoch 3/3
4/4 - 3s - loss: 1.4740 - accuracy: 0.2200 - 3s/epoch - 862ms/step
```

Out[ ]:
```
<keras.callbacks.History at 0x1c0136f71f0>
```