Amina Tabassum

HW - 04

NUID : 002190127

Problem 1

1.1 Loading dataset and building input pipeline

```
import tensorflow as tf
import tensorflow_datasets as tfds

print('tensorflow version is: ',tf.__version__)
print('datset version is : ', tfds.__version__)

#load dataset
(training_ds,validation_ds,test_ds),info=tfds.load('mnist',split=['tra
in[:90%]','train[90%:]','test'],shuffle_files=True,as_supervised=True,
with_info=True )

training_ds,validation_ds,test_ds
```

```
tensorflow version is:  2.4.1
datset version is :  4.8.3
```

```
(<PrefetchDataset shapes: ((28, 28, 1), ()), types: (tf.uint8,
tf.int64)>,
 <PrefetchDataset shapes: ((28, 28, 1), ()), types: (tf.uint8,
tf.int64)>,
 <PrefetchDataset shapes: ((28, 28, 1), ()), types: (tf.uint8,
tf.int64)>)
```

Normalize dataset

```
def normalize_img(image,label):
    image=tf.cast(image,tf.float32)
    image=image/255.0
    return image,label
```

Build input pipeline

```
batch_size=64
buffer_size=10000
```

```
#training data pipeline
training_ds=training_ds.map(normalize_img,num_parallel_calls=tf.data.A
UTOTUNE)
training_ds=training_ds.cache()
training_ds=training_ds.shuffle(buffer_size)
training_ds=training_ds.batch(batch_size)
training_ds=training_ds.prefetch(tf.data.AUTOTUNE)

#validation data pipeline
validation_ds=validation_ds.map(normalize_img,num_parallel_calls=tf.da
ta.AUTOTUNE)
validation_ds=validation_ds.batch(batch_size)
validation_ds=validation_ds.cache()
validation_ds=validation_ds.prefetch(tf.data.AUTOTUNE)

#test data pipeline
test_ds=test_ds.map(normalize_img,num_parallel_calls=tf.data.AUTOTUNE)
test_ds=test_ds.batch(batch_size)
test_ds=test_ds.cache()
test_ds=test_ds.prefetch(tf.data.AUTOTUNE)

print(training_ds.element_spec)
print(validation_ds.element_spec)
print(test_ds.element_spec)
```

```
(TensorSpec(shape=(None, 28, 28, 1), dtype=tf.float32, name=None),
TensorSpec(shape=(None,), dtype=tf.int64, name=None))
(TensorSpec(shape=(None, 28, 28, 1), dtype=tf.float32, name=None),
TensorSpec(shape=(None,), dtype=tf.int64, name=None))
(TensorSpec(shape=(None, 28, 28, 1), dtype=tf.float32, name=None),
TensorSpec(shape=(None,), dtype=tf.int64, name=None))
```

1.2 CNN network model

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, Dense, Flatten, MaxPool2D
from tensorflow.keras.callbacks import EarlyStopping

model=Sequential()
model.add(Conv2D(32,
(3,3),padding='same',strides=1,activation='relu',input_shape=(28,28,1)
))
model.add(Conv2D(64,(3,3),padding='same',strides=1,activation='relu'))
model.add(MaxPool2D())
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

1.3 Compile CNN model

```
model.compile(tf.optimizers.Adam(learning_rate=0.001,beta_1=0.9,beta_2
=0.999,epsilon=1e-
7,name='Adam'),loss='sparse_categorical_crossentropy',metrics=['accura
cy'])
```

Train CNN

```
history=model.fit(training_ds,epochs=6,validation_data=validation_ds)
```

```
Epoch 1/6
844/844 [==============================] - 58s 68ms/step - loss:
0.2913 - accuracy: 0.9116 - val_loss: 0.0556 - val_accuracy: 0.9835
Epoch 2/6
844/844 [==============================] - 57s 68ms/step - loss:
0.0457 - accuracy: 0.9856 - val_loss: 0.0454 - val_accuracy: 0.9848
Epoch 3/6
844/844 [==============================] - 59s 70ms/step - loss:
0.0249 - accuracy: 0.9922 - val_loss: 0.0439 - val_accuracy: 0.9867
Epoch 4/6
844/844 [==============================] - 58s 69ms/step - loss:
0.0173 - accuracy: 0.9943 - val_loss: 0.0425 - val_accuracy: 0.9875
Epoch 5/6
844/844 [==============================] - 59s 70ms/step - loss:
0.0138 - accuracy: 0.9951 - val_loss: 0.0489 - val_accuracy: 0.9887
Epoch 6/6
844/844 [==============================] - 59s 70ms/step - loss:
0.0092 - accuracy: 0.9971 - val_loss: 0.0477 - val_accuracy: 0.9875
```

1.4 Evaluate test-set Performance

```
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 28, 28, 32)        320
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 28, 28, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_8 (Dense)              (None, 128)               1605760
_________________________________________________________________
dense_9 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,625,866
```

```
Trainable params: 1,625,866
Non-trainable params: 0
_________________________________________________________________
```

```
testing_loss,testing_Acc=model.evaluate(test_ds)
print('test dataset accuracy is:',testing_Acc)
print('loss is', testing_loss)
```

```
157/157 [==============================] - 2s 11ms/step - loss: 0.0434
- accuracy: 0.9899
test dataset accuracy is: 0.9898999929428101
loss is 0.04341074079275131
```

Problem 2

```
import tensorflow as tf
import tensorflow_datasets as tfds

print('tensorflow version is: ',tf.__version__)
print('datset version is : ', tfds.__version__)

#load dataset
(training_ds,validation_ds,test_ds),info=tfds.load('mnist',split=['tra
in[:90%]','train[90%:]','test'],shuffle_files=True,as_supervised=True,
with_info=True )

training_ds,validation_ds,test_ds
```

```
tensorflow version is:  2.4.1
datset version is :  4.8.3
```

```
(<PrefetchDataset shapes: ((28, 28, 1), ()), types: (tf.uint8,
tf.int64)>,
 <PrefetchDataset shapes: ((28, 28, 1), ()), types: (tf.uint8,
tf.int64)>,
 <PrefetchDataset shapes: ((28, 28, 1), ()), types: (tf.uint8,
tf.int64)>)
```

```
def normalize_img(image,label):
    image=tf.cast(image,tf.float32)
    image=image/255.0
    return image,label

batch_size=64
buffer_size=10000

#training data pipeline
training_ds=training_ds.map(normalize_img,num_parallel_calls=tf.data.A
UTOTUNE)
training_ds=training_ds.cache()
training_ds=training_ds.shuffle(buffer_size)
```

```
training_ds=training_ds.batch(batch_size)
training_ds=training_ds.prefetch(tf.data.AUTOTUNE)

#validation data pipeline
validation_ds=validation_ds.map(normalize_img,num_parallel_calls=tf.da
ta.AUTOTUNE)
validation_ds=validation_ds.batch(batch_size)
validation_ds=validation_ds.cache()
validation_ds=validation_ds.prefetch(tf.data.AUTOTUNE)

#test data pipeline
test_ds=test_ds.map(normalize_img,num_parallel_calls=tf.data.AUTOTUNE)
test_ds=test_ds.batch(batch_size)
test_ds=test_ds.cache()
test_ds=test_ds.prefetch(tf.data.AUTOTUNE)

print(training_ds.element_spec)
print(validation_ds.element_spec)
print(test_ds.element_spec)
```

```
(TensorSpec(shape=(None, 28, 28, 1), dtype=tf.float32, name=None),
TensorSpec(shape=(None,), dtype=tf.int64, name=None))
(TensorSpec(shape=(None, 28, 28, 1), dtype=tf.float32, name=None),
TensorSpec(shape=(None,), dtype=tf.int64, name=None))
(TensorSpec(shape=(None, 28, 28, 1), dtype=tf.float32, name=None),
TensorSpec(shape=(None,), dtype=tf.int64, name=None))
```

```
model_mlp=Sequential()
model_mlp.add(Flatten())
model_mlp.add(Dense(300,activation='relu',input_shape=(784,)))
model_mlp.add(Dense(100,activation='relu'))
model_mlp.add(Dense(10,activation='softmax'))

model_mlp.compile(tf.optimizers.Adam(learning_rate=0.001,beta_1=0.9,be
ta_2=0.999,epsilon=1e-
7,name='Adam'),loss='sparse_categorical_crossentropy',metrics=['accura
cy'])


history_mlp=model_mlp.fit(training_ds,epochs=6,validation_data=validat
ion_ds)
```

```
Epoch 1/6
844/844 [==============================] - 2s 2ms/step - loss: 0.4293
- accuracy: 0.8759 - val_loss: 0.1171 - val_accuracy: 0.9672
Epoch 2/6
844/844 [==============================] - 2s 2ms/step - loss: 0.0995
- accuracy: 0.9709 - val_loss: 0.1003 - val_accuracy: 0.9682
Epoch 3/6
844/844 [==============================] - 2s 2ms/step - loss: 0.0643
```

```
- accuracy: 0.9804 - val_loss: 0.0867 - val_accuracy: 0.9743
Epoch 4/6
844/844 [==============================] - 2s 2ms/step - loss: 0.0449
- accuracy: 0.9861 - val_loss: 0.0804 - val_accuracy: 0.9758
Epoch 5/6
844/844 [==============================] - 2s 2ms/step - loss: 0.0361
- accuracy: 0.9883 - val_loss: 0.0790 - val_accuracy: 0.9775
Epoch 6/6
844/844 [==============================] - 2s 2ms/step - loss: 0.0282
- accuracy: 0.9909 - val_loss: 0.0800 - val_accuracy: 0.9787
```

```
model_mlp.summary()
```

```
Model: "sequential_20"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_7 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_34 (Dense)             (None, 300)               235500
_________________________________________________________________
dense_35 (Dense)             (None, 100)               30100
_________________________________________________________________
dense_36 (Dense)             (None, 10)                1010
=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
_________________________________________________________________
```

```
testing_loss,testing_Acc=model_mlp.evaluate(test_ds)
print('test dataset accuracy is:',testing_Acc)
print('loss is', testing_loss)
```

```
157/157 [==============================] - 0s 1ms/step - loss: 0.0759
- accuracy: 0.9782
test dataset accuracy is: 0.9782000184059143
loss is 0.07590688765048981
```

Problem 3

```
model_mlp_1=Sequential()
model_mlp_1.add(Flatten())
model_mlp_1.add(Dense(100,activation='relu',input_shape=(784,)))
model_mlp_1.add(Dense(10,activation='softmax'))

model_mlp_1.compile(tf.optimizers.Adam(learning_rate=0.001,beta_1=0.9,
beta_2=0.999,epsilon=1e-
7,name='Adam'),loss='sparse_categorical_crossentropy',metrics=['accura
```

```
cy'])
```

```
history_mlp_1=model_mlp_1.fit(training_ds,epochs=6,validation_data=val
idation_ds)
```

```
Epoch 1/6
844/844 [==============================] - 1s 1ms/step - loss: 0.5435
- accuracy: 0.8509 - val_loss: 0.1995 - val_accuracy: 0.9455
Epoch 2/6
844/844 [==============================] - 1s 1ms/step - loss: 0.1655
- accuracy: 0.9523 - val_loss: 0.1337 - val_accuracy: 0.9632
Epoch 3/6
844/844 [==============================] - 1s 1ms/step - loss: 0.1110
- accuracy: 0.9677 - val_loss: 0.1166 - val_accuracy: 0.9683
Epoch 4/6
844/844 [==============================] - 1s 1ms/step - loss: 0.0855
- accuracy: 0.9742 - val_loss: 0.1035 - val_accuracy: 0.9717
Epoch 5/6
844/844 [==============================] - 1s 1ms/step - loss: 0.0676
- accuracy: 0.9801 - val_loss: 0.0974 - val_accuracy: 0.9713
Epoch 6/6
844/844 [==============================] - 1s 1ms/step - loss: 0.0570
- accuracy: 0.9835 - val_loss: 0.0940 - val_accuracy: 0.9742
```

```
model_mlp_1.summary()
```

```
Model: "sequential_22"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_9 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_39 (Dense)             (None, 100)               78500
_________________________________________________________________
dense_40 (Dense)             (None, 10)                1010
=================================================================
Total params: 79,510
Trainable params: 79,510
Non-trainable params: 0
_________________________________________________________________
```

```
testing_loss,testing_Acc=model_mlp_1.evaluate(test_ds)
print('test dataset accuracy is:',testing_Acc)
print('loss is', testing_loss)
```

```
157/157 [==============================] - 0s 695us/step - loss:
0.0894 - accuracy: 0.9736
test dataset accuracy is: 0.9735999703407288
loss is 0.0894041359424591
```