Assignment 6.1

author: Rachel Nelson

class: DSC650

Assignment 5.1

Using section 5.1 in Deep Learning with Python as a guide (listing 5.3 in particular), create a ConvNet model that classifies images in the MNIST digit dataset. Save the model, predictions, metrics, and validation plots in the dsc650/assignments/assignment06/results directory. If you are using JupyterHub, you can include those plots in your Jupyter notebook.

```
4  # Import Packages
   from keras.datasets import mnist
   from tensorflow.keras.utils import to_categorical
   from keras import layers
   from keras import models
   import tensorflow as tf
```

```
7  # Instantiating a small covnet
   model = models.Sequential()
   model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
   model.add(layers.MaxPooling2D((2, 2)))
   model.add(layers.Conv2D(64, (3, 3), activation='relu'))
   model.add(layers.MaxPooling2D((2, 2)))
   model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

```
8  # Creating Summary of Model
   model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

=================================================================
Total params: 55,744
Trainable params: 55,744
```

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

```
# 5.2 Adding a classifier on top of the covnet

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

```
# Creating Summary of Model using a final layer with 10 outputs and a softmax activation
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          36928

 flatten (Flatten)           (None, 576)               0

 dense (Dense)               (None, 64)                36928

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

```
# 5.3 Training the covnet on MNIST images
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
model.compile(optimizer='rmsprop',
loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=64)
```

```
Epoch 1/5
938/938 [==============================] - 15s 15ms/step - loss: 0.1681 - accuracy: 0.9471□□□□□□□□□□□□□□□□□□
Epoch 2/5
938/938 [==============================] - 14s 15ms/step - loss: 0.0448 - accuracy: 0.9863□□□□□□□□□□□□□□□□□□
Epoch 3/5
938/938 [==============================] - 14s 15ms/step - loss: 0.0307 - accuracy: 0.9905□□□□□□□□□□□□□□□□□□
Epoch 4/5
938/938 [==============================] - 14s 15ms/step - loss: 0.0235 - accuracy: 0.9927□□□□□□□□□□□□□□□□□□
```

```
Epoch 5/5
938/938 [==============================] - 14s 15ms/step - loss: 0.0190 - accuracy: 0.9939□□□□□□□□□□□□□□□□□□
```

11
```
<keras.callbacks.History at 0x174b0d27748>
```

12
```
# Evaluate the model on test data
test_loss, test_acc = model.evaluate(test_images, test_labels)
test_acc
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.0318 - accuracy: 0.9908  □□□□□□□□□□□□□□□□□□
```

12
```
0.9908000230789185
```

13
```
model_no_max_pool = models.Sequential()
model_no_max_pool.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(28, 28, 1)))
model_no_max_pool.add(layers.Conv2D(64, (3, 3), activation='relu'))
model_no_max_pool.add(layers.Conv2D(64, (3, 3), activation='relu'))
model_no_max_pool.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 26, 26, 32)        320

 conv2d_4 (Conv2D)           (None, 24, 24, 64)        18496

 conv2d_5 (Conv2D)           (None, 22, 22, 64)        36928

=================================================================
Total params: 55,744
Trainable params: 55,744
Non-trainable params: 0
_________________________________________________________________
```