Open in Colab

In the course you learned how to do classification using Fashion MNIST, a data set containing items of clothing. There's another, similar dataset called MNIST which has items of handwriting -- the digits 0 through 9.

Write an MNIST classifier that trains to 99% accuracy or above, and does it without a fixed number of epochs -- i.e. you should stop training once you reach that level of accuracy.

Some notes:

1. It should succeed in less than 10 epochs, so it is okay to change epochs to 10, but nothing larger
2. When it reaches 99% or greater it should print out the string "Reached 99% accuracy so cancelling training!"
3. If you add any additional variables, make sure you use the same names as the ones used in the class

I've started the code for you below -- how would you finish it?

Things to Note:

1. When coding the `class myCallback`, Python 3 will run into an error

   ```
   TypeError: '>' not supported between instances of 'NoneType' and 'float'
   ```

   when using the code

   ```
   if(logs.get('accuracy')>0.99):
   ```

For Python 3, use the following equivalent code line

```
if logs.get('accuracy') is not None and logs.get('accuracy') > 0.99:
```

2. You can run the notebook using TensorFlow 2.5.0

```
import tensorflow as tf
```

```
print(tf.__version__)

     2.6.0

# mnist = tf.keras.datasets.mnist
# (x_train, y_train),(x_test, y_test) = mnist.load_data()
# import numpy as np
# np.set_printoptions(linewidth=200)
# print(x_train[4])
# print(y_train[4])
# import matplotlib.pyplot as plt
# plt.imshow(x_train[4])


# GRADED FUNCTION: train_mnist
def train_mnist():

    class myCallback(tf.keras.callbacks.Callback):
        def on_epoch_end(self, epoch, logs={}):
            if logs.get('accuracy') is not None and logs.get('accuracy') > 0.99:
                print("\nReached 99% accuracy so cancelling training!")
                self.model.stop_training = True

    mnist = tf.keras.datasets.mnist

    (x_train, y_train),(x_test, y_test) = mnist.load_data()

    mnist = tf.keras.datasets.mnist

    (x_train, y_train),(x_test, y_test) = mnist.load_data()
    x_train, x_test = x_train / 255.0, x_test / 255.0

    callbacks = myCallback()

    model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape=(28, 28)), # Adding a layer with single channel image 28x28
        tf.keras.layers.Dense(512, activation=tf.nn.relu), # Adds a layer of neurons
        tf.keras.layers.Dense(10, activation=tf.nn.softmax) # Takes a set of values and effectively picks the biggest one between 0 or 1
    ])

    model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    # model fitting
    history = model.fit(x_train, y_train, epochs=10, callbacks=[callbacks] # CallBack function check and stop epoch once the model accuracy exceeding
    )
    # model fitting

    return history.epoch, history.history['accuracy'][-1]


history.epoch, history.history['accuracy'][-1]train_mnist()
```

```
Epoch 1/10
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2003 - accuracy: 0.9414
Epoch 2/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0788 - accuracy: 0.9762
Epoch 3/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0516 - accuracy: 0.9836
Epoch 4/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0359 - accuracy: 0.9880
Epoch 5/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0265 - accuracy: 0.9913

Reached 99% accuracy so cancelling training!
([0, 1, 2, 3, 4], 0.9912833571434021)
```

0s completed at 1:53 PM