Classifying Hand-Written Numbers

Student number: 20195851 Name: Habib Md Arafat

1. There are some errors in the uploaded file, “handwrittennumber.py”. This py files would like to download the handwritten number data, and train a machine learning model, and then test the trained model. Please correct the errors to make up its initial intention. Please write down your corrected code and insert the captured result of your code.

Answer:

```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt

(train_images, train_labels), (test_images, test_labels) = datasets.mni
st.load_data()

train_images=train_images.reshape ((60000,28,28,1))
test_images=test_images.reshape ((10000,28,28,1))
train_images, test_images = train_images / 255.0, test_images / 255.0


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'))
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 3, 3, 64)          36928
=================================================================
Total params: 55,744
Trainable params: 55,744
Non-trainable params: 0
_________________________________________________________________
```

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

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_11 (Conv2D)           (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 3, 3, 64)          36928
_________________________________________________________________
flatten_6 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_12 (Dense)             (None, 64)                36928
_________________________________________________________________
dense_13 (Dense)             (None, 10)                650
=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_l
ogits=True),
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

```
Epoch 1/10
1875/1875 [==============================] - 57s 30ms/step - loss: 1.5317 - accuracy: 0.9320 - val_loss: 1.4893 - val_accuracy: 0.9727
Epoch 2/10
1875/1875 [==============================] - 57s 30ms/step - loss: 1.4846 - accuracy: 0.9771 - val_loss: 1.4770 - val_accuracy: 0.9843
Epoch 3/10
1875/1875 [==============================] - 57s 30ms/step - loss: 1.4794 - accuracy: 0.9819 - val_loss: 1.4739 - val_accuracy: 0.9882
Epoch 4/10
1875/1875 [==============================] - 56s 30ms/step - loss: 1.4769 - accuracy: 0.9845 - val_loss: 1.4786 - val_accuracy: 0.9826
Epoch 5/10
1875/1875 [==============================] - 56s 30ms/step - loss: 1.4760 - accuracy: 0.9852 - val_loss: 1.4747 - val_accuracy: 0.9865
Epoch 6/10
1875/1875 [==============================] - 56s 30ms/step - loss: 1.4745 - accuracy: 0.9867 - val_loss: 1.4754 - val_accuracy: 0.9856
Epoch 7/10
1875/1875 [==============================] - 57s 30ms/step - loss: 1.4735 - accuracy: 0.9876 - val_loss: 1.4734 - val_accuracy: 0.9875
Epoch 8/10
1875/1875 [==============================] - 56s 30ms/step - loss: 1.4738 - accuracy: 0.9873 - val_loss: 1.4745 - val_accuracy: 0.9864
Epoch 9/10
1875/1875 [==============================] - 57s 30ms/step - loss: 1.4731 - accuracy: 0.9880 - val_loss: 1.4739 - val_accuracy: 0.9870
Epoch 10/10
1875/1875 [==============================] - 56s 30ms/step - loss: 1.4731 - accuracy: 0.9880 - val_loss: 1.4780 - val_accuracy: 0.9830
```

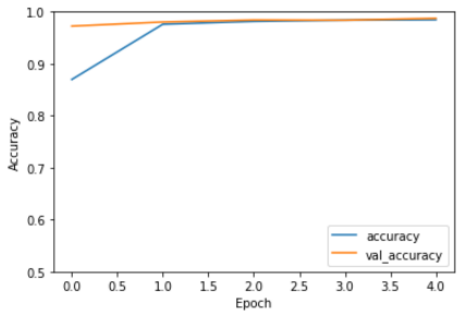
```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')
```

```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose
=2)
```

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

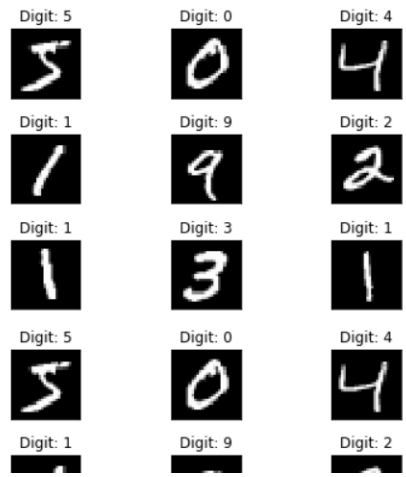
```
313/313 - 3s - loss: 1.4739 - accuracy: 0.9874
```



2. Please show first 5 images in the training images with the corresponding label. Please write down your code and insert the captured result of your code.

Answer: Images with corresponding labels

```
import matplotlib.pyplot as plt
fig = plt.figure()
for i in range(9):
  plt.subplot(3,3,i+1)
  plt.tight_layout()
  plt.imshow(X_train[i], cmap='gray', interpolation='none')
  plt.title("Digit: {}".format(y_train[i]))
  plt.xticks([])
  plt.yticks([])
fig
```



3. At your answer of the above (1), please change the kernel size as (1,1) of all the convolutional layer. Please write down your code and insert the captured result of your code.

Answer: Changed portion of the code:

```
model = models.Sequential()
model.add(layers.Conv2D(32, kernel_size=(1, 1), activation='relu', input_shape=((28,28,1))))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, kernel_size=(1, 1), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, kernel_size=(1, 1), activation='relu'))
model.summary()
```

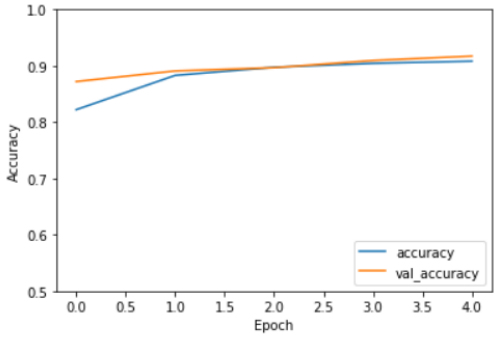
```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 28, 28, 32)        64
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 14, 14, 64)        2112
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 7, 7, 64)          4160
=================================================================
Total params: 6,336
Trainable params: 6,336
Non-trainable params: 0
```

Output:

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

```
313/313 - 1s - loss: 1.5431 - accuracy: 0.9172
```



4. At your answer of the above (1), please change the kernel size as (1,1) of all the max pooling layer. Please write down your code and insert the captured result of your code.

Answer: Changed source code.

```
model = models.Sequential()
model.add(layers.Conv2D(32, kernel_size=(1, 1), activation='relu', input_shape=((28,28,1))))
model.add(layers.MaxPooling2D((1, 1)))
model.add(layers.Conv2D(64, kernel_size=(1, 1), activation='relu'))
model.add(layers.MaxPooling2D((1, 1)))
model.add(layers.Conv2D(64, kernel_size=(1, 1), activation='relu'))
model.summary()
```

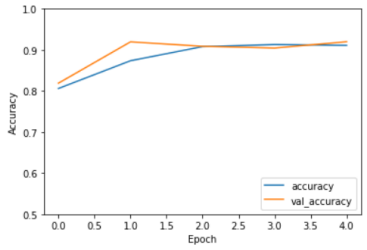
```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 28, 28, 32)        64
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 28, 28, 32)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 28, 28, 64)        2112
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 28, 28, 64)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 28, 28, 64)        4160
=================================================================
Total params: 6,336
Trainable params: 6,336
Non-trainable params: 0
```

Output:

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

```
313/313 - 5s - loss: 1.5413 - accuracy: 0.9199
```



At your answer of the above (1), please remove one set of the convolutional layer and the max pooling layer. Please write down your code and insert the captured result of your code.

Changed code:

```
model = models.Sequential()
model.add(layers.Conv2D(32, kernel_size=(1, 1), activation='relu', input_shape=((28,28,1))))
model.add(layers.MaxPooling2D((1, 1)))
model.add(layers.Conv2D(64, kernel_size=(1, 1), activation='relu'))
model.summary()
```

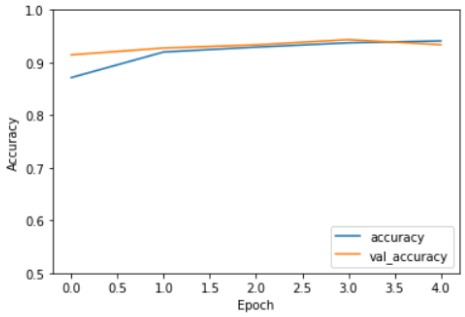
```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_21 (Conv2D)           (None, 28, 28, 32)        64
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 28, 28, 32)        0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 28, 28, 64)        2112
=================================================================
Total params: 2,176
Trainable params: 2,176
Non-trainable params: 0
_________________________________________________________________
```

Output:

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

```
313/313 - 3s - loss: 1.5274 - accuracy: 0.9338
```



5. Using the best model among the above, classify 10 numbers in the uploaded file, “numbers4test.png”. Please write down your code and insert the captured result of your code.

Answer: I did not find the numbers4Test.png file. However, classified numbers with output are given below. The model with the highest accuracy has been used.

```
y_pred = model.predict(test_images)
X_test__ = test_images.reshape(test_images.shape[0], 28, 28)

fig, axis = plt.subplots(4, 4, figsize=(12, 14))
for i, ax in enumerate(axis.flat):
    ax.imshow(X_test__[i], cmap='binary')
    ax.set(title = f"Predict Number is {y_pred[i].argmax()}");
```