This example is similar to, although not identical to, examples given in Chapter 7 of the book Deep Learning with Python, Second Edition.

Convolutional Neural Network example

This second example looks at generating a convolutional neural network. Again, its function is both to investigate the network topology and it abailities but also to familiarise you with how to construct such a network in KERAS.

This example will again use the functional API components of KERAS

The netowk will again use a character recognition task using the standard MNIST dataset.

Defining the network architecture

As for the first network, this is the first section you need to write yourselves.

The workshop script takes you through what you need to do. Note again the Inputs are defined for you however. Your function is to fill in the missing sections to define the layers you will need to employ. Naturally in this section you will need to define convolutional, pooling and flattening layer (the latter is already included).

In [1]:

```
from tensorflow import keras
from tensorflow.keras import layers

inputs = keras.Input(shape=(28, 28, 1))
# First convolutional layer
x = layers.Conv2D(32, (3, 3), activation="relu", padding="same")(in
x = layers.MaxPooling2D(pool_size=(2, 2))(x)

# Second convolutional layer
x = layers.Conv2D(64, (3, 3), activation="relu", padding="same")(x)
x = layers.MaxPooling2D(pool_size=(2, 2))(x)

# Third convolutional layer (newly added)
x = layers.Conv2D(128, (3, 3), activation="relu", padding="same")(x
x = layers.MaxPooling2D(pool_size=(2, 2))(x)

# Flatten the output before feeding into Dense layers
x = layers.Flatten()(x)
x = layers.Dense(128, activation="relu")(x)

# Output layer (10 classes for digits)
outputs = layers.Dense(10, activation="softmax")(x)  # 10 classes f
```

```
model = keras.Model(inputs=inputs, outputs=outputs)
```

```
2025-02-10 12:43:47.226394: E tensorflow/compiler/xla/stream_executo
r/cuda/cuda_dnn.cc:9342] Unable to register cuDNN factory: Attemptin
g to register factory for plugin cuDNN when one has already been reg
istered
2025-02-10 12:43:47.226507: E tensorflow/compiler/xla/stream_executo
r/cuda/cuda_fft.cc:609] Unable to register cuFFT factory: Attempting
to register factory for plugin cuFFT when one has already been regis
tered
2025-02-10 12:43:47.226558: E tensorflow/compiler/xla/stream_executo
r/cuda/cuda_blas.cc:1518] Unable to register cuBLAS factory: Attempt
ing to register factory for plugin cuBLAS when one has already been
registered
2025-02-10 12:43:48.802661: W tensorflow/compiler/tf2tensorrt/utils/
py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
```

Displaying the model's summary

In [2]:
```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 28, 28, 1)]       0

 conv2d (Conv2D)             (None, 28, 28, 32)        320

 max_pooling2d (MaxPooling2  (None, 14, 14, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 14, 14, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 7, 7, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (None, 7, 7, 128)         73856

 max_pooling2d_2 (MaxPoolin  (None, 3, 3, 128)         0
 g2D)

 flatten (Flatten)           (None, 1152)              0

 dense (Dense)               (None, 128)               147584

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 241546 (943.54 KB)
Trainable params: 241546 (943.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Compiling and training the model

The next stage is to compile the modeul using an optimiser and an error calculation. Follow the script to deduce what to put here. The images are preprocessed for you as for the first example.

Train the model analogously to the first example.

In [3]:
```
from tensorflow.keras.datasets import mnist

(train_images, train_labels), (test_images, test_labels) = mnist.lo
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

model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])

model.fit(train_images, train_labels, epochs=5, batch_size=32, vali
```

```
Epoch 1/5
1875/1875 [==============================] - 47s 25ms/step - loss:
0.1272 - accuracy: 0.9602 - val_loss: 0.0465 - val_accuracy: 0.9850
Epoch 2/5
1875/1875 [==============================] - 39s 21ms/step - loss:
0.0404 - accuracy: 0.9875 - val_loss: 0.0432 - val_accuracy: 0.9863
Epoch 3/5
1875/1875 [==============================] - 37s 20ms/step - loss:
0.0297 - accuracy: 0.9910 - val_loss: 0.0264 - val_accuracy: 0.9913
Epoch 4/5
1875/1875 [==============================] - 36s 19ms/step - loss:
0.0217 - accuracy: 0.9931 - val_loss: 0.0266 - val_accuracy: 0.9911
Epoch 5/5
1875/1875 [==============================] - 42s 22ms/step - loss:
0.0175 - accuracy: 0.9945 - val_loss: 0.0292 - val_accuracy: 0.9916
```

Out[3]:
```
<keras.src.callbacks.History at 0x7ff70f06faf0>
```

Evaluating the convnet

In [4]:
```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f"Test accuracy: {test_acc:.3f}")
```

```
313/313 [==============================] - 3s 10ms/step - loss: 0.02
92 - accuracy: 0.9916
Test accuracy: 0.992
```

Model Type	Architecture	Strengths Weaknesses	
MLP (Dense layers only)	3 fully connected layers	Easier to understand, works for simple problems	Struggles with image data, ignores spatial features
CNN (Conv + Pool +	3 Conv layers + Pooling +	Recognizes spatial structures, better for	More complex, needs more computation