In [1]:

```
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
```

2 Prepare the data# Prepare the data

In [2]:

```
# Model / data parameters
num_classes = 10
input_shape = (28, 28, 1)

##from tensorflow.keras.datasets import mnist
#(x_train, y_train), (x_test, y_test) = mnist.load_data()

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

# Scale images to the [0, 1] range
x_train = x_train.astype("float32") / 255
x_test = x_test.astype("float32") / 255
# Make sure images have shape (28, 28, 1)
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)
print("x_train shape:", x_train.shape)
print(x_train.shape[0], "train samples")
print(x_test.shape[0], "test samples")


# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

Build the model

In [3]:

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)

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 (MaxPooling2 (None, 5, 5, 64)          0
_____________________________________________________________
____
flatten (Flatten)            (None, 1600)              0
_____________________________________________________________
____
dropout (Dropout)            (None, 1600)              0
_____________________________________________________________
____
dense (Dense)                (None, 10)                16010
=============================================================
====
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
_____________________________________________________________
____
```

Train the model

In [4]:

```
batch_size = 128
epochs = 15

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

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validati
on_split=0.1)
```

```
Train on 54000 samples, validate on 6000 samples
Epoch 1/15
54000/54000 [==============================] - 39s 721us/samp
le - loss: 0.3668 - accuracy: 0.8877 - val_loss: 0.0801 - val
_accuracy: 0.9792
Epoch 2/15
```

```
54000/54000 [==============================] - 37s 688us/samp
le - loss: 0.1070 - accuracy: 0.9673 - val_loss: 0.0528 - val
_accuracy: 0.9850
Epoch 3/15
54000/54000 [==============================] - 34s 628us/samp
le - loss: 0.0811 - accuracy: 0.9749 - val_loss: 0.0447 - val
_accuracy: 0.9878
Epoch 4/15
54000/54000 [==============================] - 35s 643us/samp
le - loss: 0.0669 - accuracy: 0.9792 - val_loss: 0.0416 - val

_accuracy: 0.9892
Epoch 5/15
54000/54000 [==============================] - 35s 656us/samp
le - loss: 0.0602 - accuracy: 0.9808 - val_loss: 0.0379 - val
_accuracy: 0.9902
Epoch 6/15
54000/54000 [==============================] - 35s 645us/samp
le - loss: 0.0540 - accuracy: 0.9830 - val_loss: 0.0360 - val
_accuracy: 0.9903
Epoch 7/15
54000/54000 [==============================] - 34s 632us/samp
le - loss: 0.0494 - accuracy: 0.9845 - val_loss: 0.0343 - val
_accuracy: 0.9908
Epoch 8/15
54000/54000 [==============================] - 35s 644us/samp
le - loss: 0.0463 - accuracy: 0.9854 - val_loss: 0.0338 - val
_accuracy: 0.9907
Epoch 9/15
54000/54000 [==============================] - 43s 799us/samp
le - loss: 0.0421 - accuracy: 0.9864 - val_loss: 0.0302 - val
_accuracy: 0.9922
Epoch 10/15
54000/54000 [==============================] - 37s 684us/samp
le - loss: 0.0414 - accuracy: 0.9867 - val_loss: 0.0340 - val
_accuracy: 0.9907
Epoch 11/15
54000/54000 [==============================] - 37s 679us/samp
le - loss: 0.0390 - accuracy: 0.9877 - val_loss: 0.0298 - val
_accuracy: 0.9917
Epoch 12/15
54000/54000 [==============================] - 36s 660us/samp
le - loss: 0.0372 - accuracy: 0.9883 - val_loss: 0.0300 - val
_accuracy: 0.9918
Epoch 13/15
54000/54000 [==============================] - 35s 655us/samp
le - loss: 0.0345 - accuracy: 0.9888 - val_loss: 0.0308 - val
_accuracy: 0.9910
Epoch 14/15
54000/54000 [==============================] - 35s 644us/samp
le - loss: 0.0337 - accuracy: 0.9890 - val_loss: 0.0303 - val
_accuracy: 0.9920
Epoch 15/15
54000/54000 [==============================] - 34s 637us/samp
le - loss: 0.0330 - accuracy: 0.9896 - val_loss: 0.0305 - val
_accuracy: 0.9920
```

Out[4]:

```
<tensorflow.python.keras.callbacks.History at 0x28e67cd2cc8>
```

Evaluate the trained model

In [ ]:

```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```