```
"""
Title: Simple MNIST convnet
Description: A simple convnet that achieves ~99% test accuracy on MNIST.
Author: [fchollet](https://twitter.com/fchollet)
"""
```

```
'\nTitle: Simple MNIST convnet\nDescription: A simple convnet that achieves ~
llet)\n'
```

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


#Prepare the data


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


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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

```
# 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")
```

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


#Build the model


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

     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


    batch_size·=·128
    epochs·=·15

    model.compile(loss="categorical_crossentropy",·optimizer="adam",·metrics=["accurac

    model.fit(x_train,·y_train,·batch_size=batch_size,·epochs=epochs,·validation_split=
```

```
    Epoch 1/15
    422/422 [==============================] - 46s 106ms/step - loss: 0.3748 - ac
    Epoch 2/15
    422/422 [==============================] - 44s 105ms/step - loss: 0.1144 - ac
    Epoch 3/15
    422/422 [==============================] - 44s 105ms/step - loss: 0.0866 - ac
```

```
Epoch 4/15
422/422 [==============================] - 44s 105ms/step - loss: 0.0726 - ac
Epoch 5/15
422/422 [==============================] - 44s 105ms/step - loss: 0.0635 - ac
Epoch 6/15
422/422 [==============================] - 45s 106ms/step - loss: 0.0591 - ac
Epoch 7/15
422/422 [==============================] - 45s 106ms/step - loss: 0.0523 - ac
Epoch 8/15
422/422 [==============================] - 45s 105ms/step - loss: 0.0479 - ac
Epoch 9/15
422/422 [==============================] - 44s 105ms/step - loss: 0.0450 - ac
Epoch 10/15
422/422 [==============================] - 44s 105ms/step - loss: 0.0439 - ac
Epoch 11/15
422/422 [==============================] - 44s 105ms/step - loss: 0.0400 - ac
Epoch 12/15
422/422 [==============================] - 44s 105ms/step - loss: 0.0393 - ac
Epoch 13/15
422/422 [==============================] - 44s 104ms/step - loss: 0.0376 - ac
Epoch 14/15
422/422 [==============================] - 44s 104ms/step - loss: 0.0353 - ac
Epoch 15/15
422/422 [==============================] - 44s 104ms/step - loss: 0.0325 - ac
<keras.callbacks.History at 0x7fd65545c190>
```

```
#Evaluate·the·trained·model


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

```
Test loss: 0.026175467297434807
Test accuracy: 0.9901000261306763
```