```
import datetime
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

from tensorflow.keras import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.losses import categorical_crossentropy
from tensorflow.keras.layers import Dense, Flatten, Conv2D, AveragePooling2D

from tensorflow.keras import datasets
from tensorflow.keras.utils import to_categorical

from __future__ import absolute_import, division, print_function, unicode_literals


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


print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
print(x_train[0].shape, 'image shape')
```

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

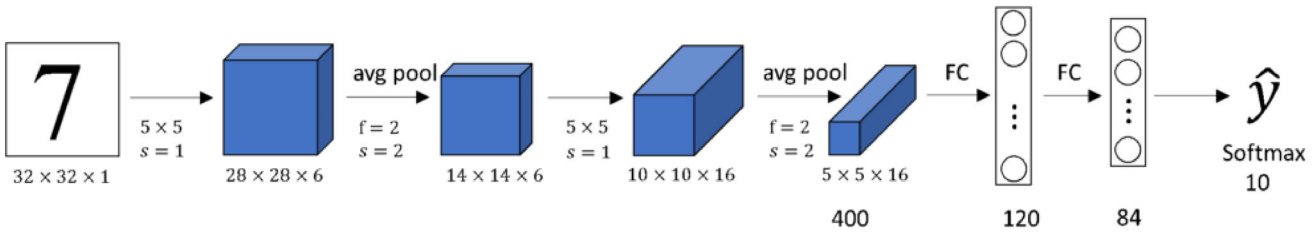
```
# Agregar una dimension
x_train = x_train[:, :, :, np.newaxis]
x_test = x_test[:, :, :, np.newaxis]

print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
print(x_train[0].shape, 'image shape')
```

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

```
# convertir clases a matrices
num_classes = 10
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)


# Normalización
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
```




```
# LeNet-5 model
class LeNet(Sequential):
  def __init__(self, input_shape, nb_classes):
    super().__init__()

    self.add(Conv2D(6, kernel_size=(5, 5), strides=(1, 1), activation='tanh', input_shape=input_shape, padding="same"))
    self.add(AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='valid'))
    self.add(Conv2D(16, kernel_size=(5, 5), strides=(1, 1), activation='tanh', padding='valid'))
    self.add(AveragePooling2D(pool_size=(2, 2), strides=(2, 2), padding='valid'))
    self.add(Flatten())
    self.add(Dense(120, activation='tanh'))
    self.add(Dense(84, activation='tanh'))
    self.add(Dense(nb_classes, activation='softmax'))
```

```
        self.compile(optimizer='adam',
                     loss=categorical_crossentropy,
                     metrics=['accuracy'])


model = LeNet(x_train[0].shape, num_classes)


model.summary()
```

```
Model: "le_net_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 28, 28, 6)         156

 average_pooling2d_4 (Averag  (None, 14, 14, 6)        0
 ePooling2D)

 conv2d_6 (Conv2D)           (None, 10, 10, 16)        2416

 average_pooling2d_5 (Averag  (None, 5, 5, 16)         0
 ePooling2D)

 flatten_2 (Flatten)         (None, 400)               0

 dense_6 (Dense)             (None, 120)               48120

 dense_7 (Dense)             (None, 84)                10164

 dense_8 (Dense)             (None, 10)                850

=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(x_train, y=y_train,
          epochs=5,
          validation_data=(x_test, y_test),
          verbose=0)
```

```
<keras.callbacks.History at 0x7f3f562f6260>
```

```
class_names = ['Cero', 'Uno', 'Dos', 'Tres', 'Cuatro',
               'Cinco', 'Seis', 'Siete', 'Ocho', 'Nueve']

y_pred = model.predict(x_test)
prediction_values=np.argmax(y_pred,axis=1)


fig = plt.figure(figsize=(15, 7))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

for i in range(50):
  ax = fig.add_subplot(5, 10, i + 1, xticks=[], yticks=[])
  ax.imshow(x_test[i,:].reshape((28,28)),cmap=plt.cm.gray_r, interpolation='nearest')

  if prediction_values[i] == np.argmax(y_test[i]):
    ax.text(0, 7, class_names[prediction_values[i]], color='blue')
  else:
    ax.text(0, 7, class_names[prediction_values[i]], color='red')
```

```
313/313 [==============================] - 1s 2ms/step
```

Actividad: usar el modelo LeNet previo para resolver el problema de classificación de prendas mostrado a continuación.

```
(x_train_, y_train_), (x_test_, y_test_) = datasets.fashion_mnist.load_data()

x_train_ = x_train_[:, :, :, np.newaxis]
x_test_ = x_test_[:, :, :, np.newaxis]

y_train_ = to_categorical(y_train_, num_classes)
y_test_ = to_categorical(y_test_, num_classes)

x_train_ = x_train_.astype('float32')
x_test_ = x_test_.astype('float32')
x_train_ /= 255
x_test_ /= 255

model_ = LeNet(x_train_[0].shape, num_classes)

model_.fit(x_train_, y = y_train_,
           epochs=5,
           validation_data=(x_test_, y_test_),
           verbose=0)

class_names = ['Camiseta/top', 'Pantalones', 'Jersey', 'Vestido', 'Abrigo',
               'Sandalia', 'Camisa', 'Sneaker', 'Bolso', 'Botines']

y_pred_ = model_.predict(x_test_)
prediction_values_ = np.argmax(y_pred_,axis=1)

fig = plt.figure(figsize=(15, 7))
fig.subplots_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)

for i in range(40):
  ax = fig.add_subplot(5, 8, i + 1, xticks=[], yticks=[])
  ax.imshow(x_test_[i,:].reshape((28,28)), cmap=plt.cm.gray_r, interpolation='nearest')

  if prediction_values_[i] == np.argmax(y_test_[i]):
    ax.text(0, 7, class_names[prediction_values_[i]], color='blue')
  else:
    ax.text(0, 7, class_names[prediction_values_[i]], color='red')
```

```
313/313 [==============================] - 1s 2ms/step
```

Productos de pago de Colab - Cancelar contratos

✓ 45 s completado a las 22:41