Tarea 3

September 6, 2025

1 Perceptrón Multicapa (MLP) para clasificación de imágenes

En esta libreta aprenderemos los pasos para implementar un clasificador (discriminador) de imágenes (las del conjunto de datos MNIST), usando un perceptrón multicapa y Keras.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras import layers, models, optimizers, utils, datasets
```

1.1 1. Preparar los datos

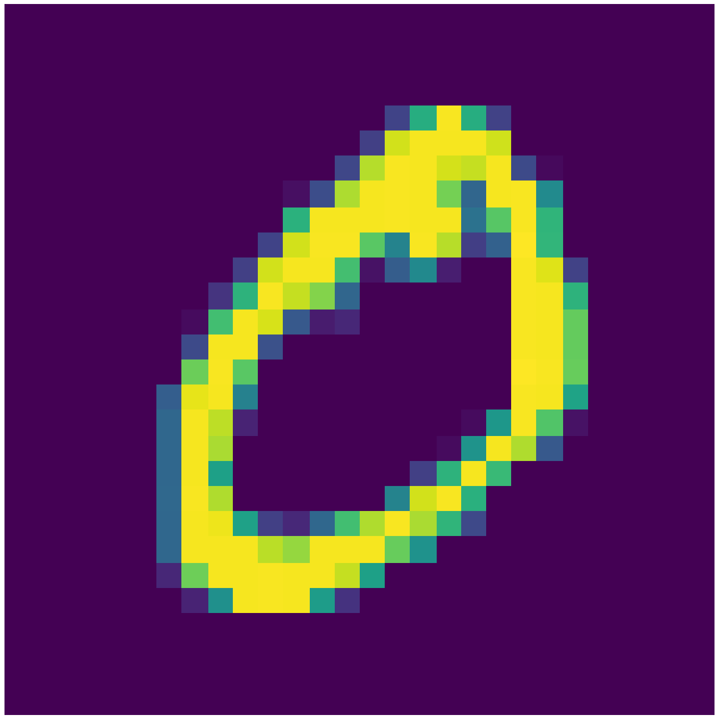
1.1 Descargar el dataset

[2]:
```
(x_train, y_train), (x_test, y_test) = datasets.mnist.load_data()
```

[3]:
```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(60000, 28, 28)
(60000,)
(10000, 28, 28)
(10000,)
```

[4]:
```
plt.imshow(x_train[1])
plt.axis('off')
plt.show()
```



1.2 Escalar los valores de las imágenes

[5]:
```python
# Normalizar el dataset

x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0
```

[6]:
```python
x_train[1]
```

[6]:
```
array([[0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        ],
       [0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
```

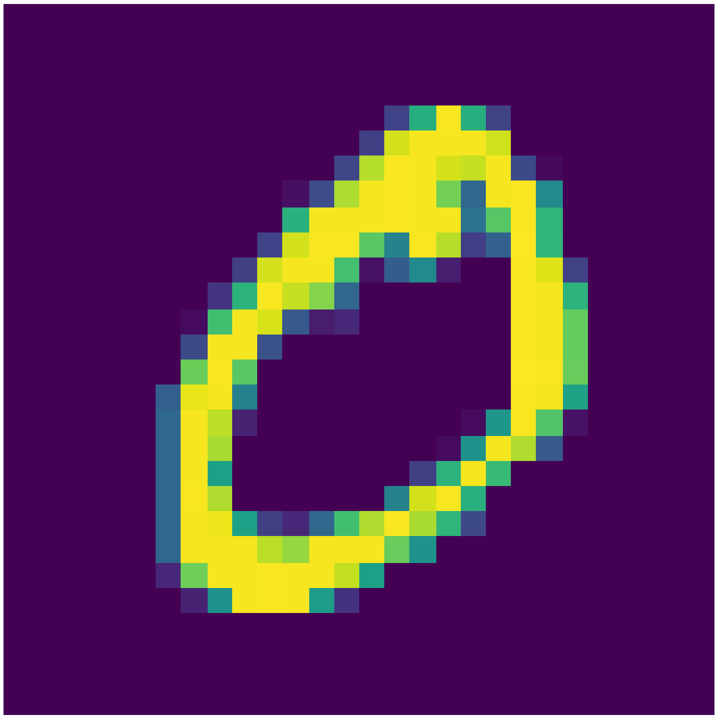
```
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.2       , 0.62352943, 0.99215686, 0.62352943, 0.19607843,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.1882353 ,
 0.93333334, 0.9882353 , 0.9882353 , 0.9882353 , 0.92941177,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.21176471, 0.8901961 ,
 0.99215686, 0.9882353 , 0.9372549 , 0.9137255 , 0.9882353 ,
 0.22352941, 0.02352941, 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.03921569, 0.23529412, 0.8784314 , 0.9882353 ,
 0.99215686, 0.9882353 , 0.7921569 , 0.32941177, 0.9882353 ,
 0.99215686, 0.47843137, 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.6392157 , 0.9882353 , 0.9882353 , 0.9882353 ,
 0.99215686, 0.9882353 , 0.9882353 , 0.3764706 , 0.7411765 ,
 0.99215686, 0.654902  , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.2       , 0.93333334, 0.99215686, 0.99215686, 0.74509805,
 0.44705883, 0.99215686, 0.89411765, 0.18431373, 0.30980393,
 1.        , 0.65882355, 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.1882353 ,
```

```
 0.93333334, 0.9882353 , 0.9882353 , 0.7019608 , 0.04705882,
 0.29411766, 0.4745098 , 0.08235294, 0.        , 0.        ,
 0.99215686, 0.9529412 , 0.19607843, 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.14901961, 0.64705884,
 0.99215686, 0.9137255 , 0.8156863 , 0.32941177, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.99215686, 0.9882353 , 0.64705884, 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.02745098, 0.69803923, 0.9882353 ,
 0.9411765 , 0.2784314 , 0.07450981, 0.10980392, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.99215686, 0.9882353 , 0.7647059 , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.22352941, 0.9882353 , 0.9882353 ,
 0.24705882, 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.99215686, 0.9882353 , 0.7647059 , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.7764706 , 0.99215686, 0.74509805,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 1.        , 0.99215686, 0.76862746, 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.29803923, 0.9647059 , 0.9882353 , 0.4392157 ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.99215686, 0.9882353 , 0.5803922 , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.33333334, 0.9882353 , 0.9019608 , 0.09803922,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.02745098, 0.5294118 ,
 0.99215686, 0.7294118 , 0.04705882, 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.33333334, 0.9882353 , 0.8745098 , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.02745098, 0.5137255 , 0.9882353 ,
 0.88235295, 0.2784314 , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
 0.        , 0.33333334, 0.9882353 , 0.5686275 , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.1882353 , 0.64705884, 0.9882353 , 0.6784314 ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.3372549 , 0.99215686, 0.88235295, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.44705883, 0.93333334, 0.99215686, 0.63529414, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.33333334, 0.9882353 , 0.9764706 , 0.57254905,
 0.1882353 , 0.11372549, 0.33333334, 0.69803923, 0.88235295,
 0.99215686, 0.8745098 , 0.654902  , 0.21960784, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.33333334, 0.9882353 , 0.9882353 , 0.9882353 ,
 0.8980392 , 0.84313726, 0.9882353 , 0.9882353 , 0.9882353 ,
 0.76862746, 0.50980395, 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.10980392, 0.78039217, 0.9882353 , 0.9882353 ,
 0.99215686, 0.9882353 , 0.9882353 , 0.9137255 , 0.5686275 ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.09803922, 0.5019608 , 0.9882353 ,
 0.99215686, 0.9882353 , 0.5529412 , 0.14509805, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
```

```
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ]], dtype=float32)
```

[7]:
```
plt.imshow(x_train[1])
plt.axis('off')
plt.show()
```



1.3 Codificar las etiquetas con one-hot-encoding.

[8]:
```
# To categorical

y_train = utils.to_categorical(y_train, num_classes=10)
y_test = utils.to_categorical(y_test, num_classes=10)
```

Veamos las primeras 10 imágenes del conjunto de entrenamiento junto con sus etiquetas codificadas usando one-hot encoding.

[9]:
```
y_train[:10]
```

[9]:
```
array([[0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0.]])
```

1.2 2. Construir el modelo

[10]:
```
# Clasificador de CIFAR10

model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28, 1)))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

```
/Users/roicort/GitHub/PCIC/GenAI/.venv/lib/python3.13/site-
packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

Podemos utilizar el método model.summary() para inspeccionar la forma de la red en cada capa.

[11]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 784)	0
dense (Dense)	(None, 512)	401,920
dense_1 (Dense)	(None, 128)	65,664
dense_2 (Dense)	(None, 10)	1,290

Total params: 468,874 (1.79 MB)

Trainable params: 468,874 (1.79 MB)

Non-trainable params: 0 (0.00 B)

1.3 3. Compilar el modelo

En este paso, compilamos el modelo con un optimizador y una función de pérdida. También pasamos al método compile del modelo un parámetro metrics donde podemos especificar cualquier métrica adicional que nos gustaría reportar durante el entrenamiento, como el accuracy.

[12]:
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

1.4 4. Entrenar el modelo

[13]:
```
model.fit(x_train, y_train, epochs=10, batch_size=64, validation_data=(x_test,␣
↪y_test))
```

```
Epoch 1/10
938/938              2s 2ms/step -
accuracy: 0.9384 - loss: 0.2081 - val_accuracy: 0.9599 - val_loss: 0.1222
Epoch 2/10
938/938              2s 2ms/step -
accuracy: 0.9747 - loss: 0.0806 - val_accuracy: 0.9735 - val_loss: 0.0841
Epoch 3/10
938/938              2s 2ms/step -
accuracy: 0.9832 - loss: 0.0527 - val_accuracy: 0.9734 - val_loss: 0.0798
Epoch 4/10
938/938              2s 2ms/step -
accuracy: 0.9870 - loss: 0.0395 - val_accuracy: 0.9770 - val_loss: 0.0781
Epoch 5/10
938/938              2s 2ms/step -
accuracy: 0.9901 - loss: 0.0304 - val_accuracy: 0.9734 - val_loss: 0.0882
Epoch 6/10
938/938              2s 2ms/step -
accuracy: 0.9921 - loss: 0.0248 - val_accuracy: 0.9804 - val_loss: 0.0655
Epoch 7/10
938/938              2s 2ms/step -
accuracy: 0.9939 - loss: 0.0177 - val_accuracy: 0.9782 - val_loss: 0.0861
Epoch 8/10
938/938              2s 2ms/step -
```

```
accuracy: 0.9941 - loss: 0.0170 - val_accuracy: 0.9833 - val_loss: 0.0707
Epoch 9/10
938/938              2s 2ms/step -
accuracy: 0.9942 - loss: 0.0165 - val_accuracy: 0.9771 - val_loss: 0.0913
Epoch 10/10
938/938              2s 2ms/step -
accuracy: 0.9956 - loss: 0.0136 - val_accuracy: 0.9789 - val_loss: 0.0956
```

[13]:
```
<keras.src.callbacks.history.History at 0x15d2c7230>
```

1.5 5. Evaluar el modelo

Hasta ahora sabemos que el modelo tiene un desempeño del 99% en el conjunto de entrenamiento, pero ¿cómo se desempeña con imágenes que no ha visto? Para contestar esta pregunta, podemos usar el método evaluate que provee Keras.

[14]:
```
model.evaluate(x_test, y_test)
```

```
313/313              0s 522us/step -
accuracy: 0.9789 - loss: 0.0956
```

[14]:
```
[0.09557777643203735, 0.9789000153541565]
```

La salida es una lista de las métricas que estamos monitoreando: entropía cruzada categórica y precisión. Podemos observar que la precisión del modelo es aún del 97% incluso en imágenes que nunca ha visto antes. Cabe destacar que, si el modelo adivinara al azar, lograría aproximadamente un 10% de precisión (porque hay 10 clases), por lo que un 97% es excelente resultado.

Podemos observar algunas de las predicciones en el conjunto de prueba utilizando el método predict: 1.- preds es un arreglo de forma [10000, 10], es decir, un vector de 10 probabilidades de clase para cada observación. 2.-Convertimos este arreglo de probabilidades en una única predicción utilizando la función argmax de numpy. Aquí, axis = -1 le indica a la función que colapse el arreglo sobre la última dimensión (la dimensión de las clases), de modo que la forma de preds_single sea entonces [10000, 1]. 3.-actual_single contiene la etiqueta real esperada.

[15]:
```
CLASSES = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

preds = model.predict(x_test)
preds_single = CLASSES[np.argmax(preds, axis=-1)]
actual_single = CLASSES[np.argmax(y_test, axis=-1)]
```

```
313/313              0s 461us/step
```

Podemos visualizar algunas de las imágenes junto con sus etiquetas y predicciones utilizando el siguiente código:

[16]:
```
n_to_show = 10
indices = np.random.choice(range(len(x_test)), n_to_show)

fig = plt.figure(figsize=(15, 3))
fig.subplots_adjust(hspace=0.4, wspace=0.4)
```

```
for i, idx in enumerate(indices):
    img = x_test[idx]
    ax = fig.add_subplot(1, n_to_show, i + 1)
    ax.axis("off")
    ax.text(
        0.5,
        -0.35,
        "pred = " + str(preds_single[idx]),
        fontsize=10,
        ha="center",
        transform=ax.transAxes,
    )
    ax.text(
        0.5,
        -0.7,
        "act = " + str(actual_single[idx]),
        fontsize=10,
        ha="center",
        transform=ax.transAxes,
    )
    ax.imshow(img)
```