CÓDIGO ORIGINAL

import keras

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.metrics import confusion\_matrix, classification\_report, f1\_score

import seaborn as sns

# Cargar el conjunto de datos MNIST

mnist = keras.datasets.mnist

(training\_images, training\_labels), (test\_images, test\_labels) = mnist.load\_data()

# Mostrar una imagen de ejemplo

print("MOSTRAR LA IMAGEN DE EJEMPLO")

print("")

index = 1

np.set\_printoptions(linewidth=320)

print(f'Label: {training\_labels[index]}')

print(f'Image:\n {training\_images[index]}')

plt.imshow(training\_images[index])

# Normalizar las imágenes

training\_images = training\_images / 255.0

test\_images = test\_images / 255.0

# Definir y compilar el modelo

model = keras.models.Sequential([

keras.layers.Flatten(input\_shape=(28, 28)),

keras.layers.Dense(128, activation='relu'),

keras.layers.Dense(10, activation='softmax')

])

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

model.summary()

# Entrenar el modelo

history = model.fit(training\_images, training\_labels, epochs=10) #CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 12

# Graficar la historia del entrenamiento

print("GRAFICAR LA HISTORIA DEL ENTRENAMIENTO")

print("")

pd.DataFrame(history.history).plot(grid=True)

plt.show()

# Evaluar el modelo en el conjunto de entrenamiento

train\_loss, train\_accuracy = model.evaluate(training\_images, training\_labels)

print("Pérdida en el conjunto de entrenamiento:", train\_loss)

print("Precisión en el conjunto de entrenamiento:", train\_accuracy)

# Evaluar el modelo en el conjunto de prueba

test\_loss, test\_accuracy = model.evaluate(test\_images, test\_labels)

print("Pérdida en el conjunto de prueba:", test\_loss)

print("Precisión en el conjunto de prueba:", test\_accuracy)

# Predecir las etiquetas del conjunto de prueba

predictions = model.predict(test\_images)

predicted\_labels = np.argmax(predictions, axis=1)

# Matriz de confusión

print("MATRIZ DE CONFUSIÓN")

print("")

conf\_matrix = confusion\_matrix(test\_labels, predicted\_labels)

print("Matriz de confusión:\n", conf\_matrix) # Aquí se genera la matriz de confusión

# Reporte de clasificación

print( "REPORTE DE CLASIFICACIÓN")

print("")

class\_report = classification\_report(test\_labels, predicted\_labels)

print("Reporte de clasificación:\n", class\_report) # Aquí se genera el reporte de clasificación

# Cálculo del F1-score

print("CÁLCULO DEL F1-SCORE")

print("")

f1 = f1\_score(test\_labels, predicted\_labels, average='weighted')

print("F1-score:", f1) # Aquí se calcula el F1-score

# Graficar la matriz de confusión

print("GRAFICAR LA MATRIZ DE CONFUSIÓN")

print("")

plt.figure(figsize=(10, 7))

sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues")

plt.ylabel('Etiqueta verdadera')

plt.xlabel('Etiqueta predicha')

plt.title('Matriz de Confusión')

plt.show() # Aquí se grafica la matriz de confusión

# Histograma de la distribución del error

print("HISTOGRAMA DE LA DISTRIBUCIÓN DEL ERROR")

print("")

errors = test\_labels - predicted\_labels

plt.hist(errors, bins=30)

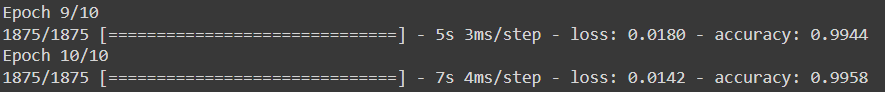
plt.xlabel('Error')

plt.ylabel('Frecuencia')

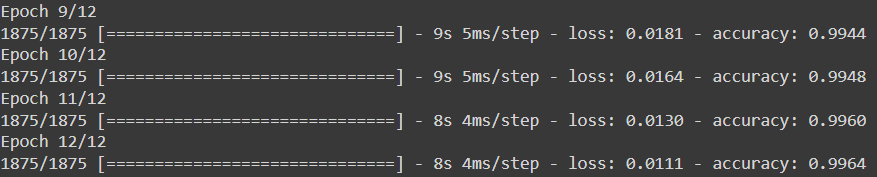
plt.title('Distribución del error')

plt.show() # Aquí se grafica la distribución del error en un histograma

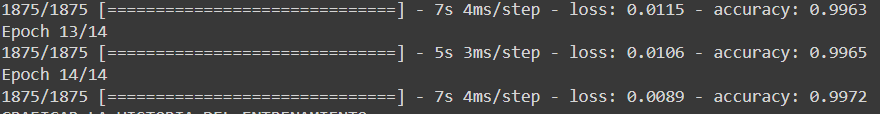
RESULTADO DE 10 EPOCHES



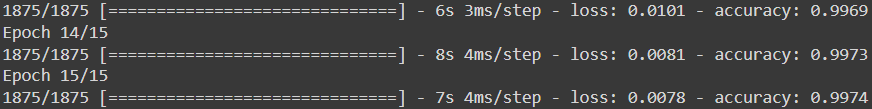
CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 12



CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 14

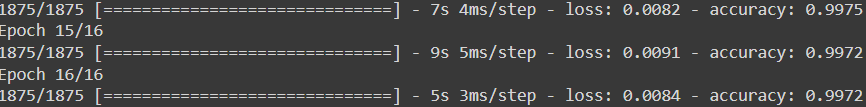


CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 15

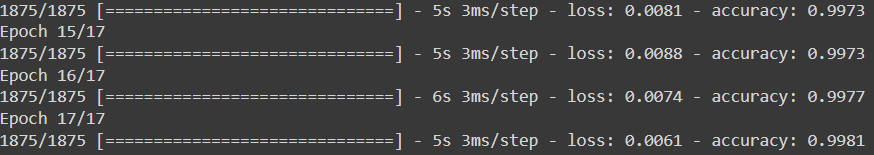


sobreajuste

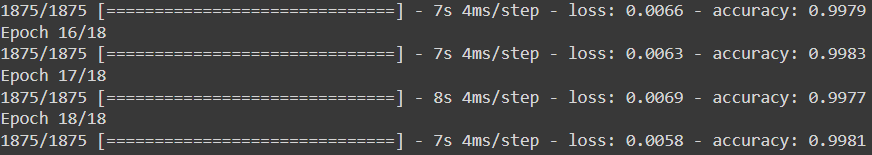
CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 16



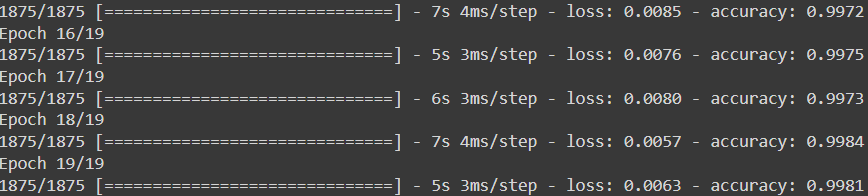
CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 17



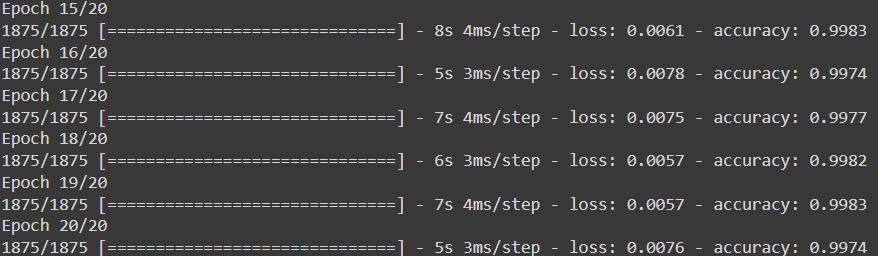
CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 18



CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 19



CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 20



Variar el número de capas

import keras

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.metrics import confusion\_matrix, classification\_report, f1\_score

import seaborn as sns

# Cargar el conjunto de datos MNIST

mnist = keras.datasets.mnist

(training\_images, training\_labels), (test\_images, test\_labels) = mnist.load\_data()

# Mostrar una imagen de ejemplo

print("MOSTRAR LA IMAGEN DE EJEMPLO")

print("")

index = 1

np.set\_printoptions(linewidth=320)

print(f'Label: {training\_labels[index]}')

print(f'Image:\n {training\_images[index]}')

plt.imshow(training\_images[index])

# Normalizar las imágenes

training\_images = training\_images / 255.0

test\_images = test\_images / 255.0

# Definir y compilar el modelo

model = keras.models.Sequential([

keras.layers.Flatten(input\_shape=(28, 28)),

keras.layers.Dense(256, activation='relu'),

keras.layers.Dense(128, activation='relu'),

keras.layers.Dense(128, activation='relu'),

keras.layers.Dense(64, activation='relu'),

keras.layers.Dense(10, activation='softmax')

])

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

model.summary()

# Entrenar el modelo

history = model.fit(training\_images, training\_labels, epochs=10) #CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 12

# Graficar la historia del entrenamiento

print("GRAFICAR LA HISTORIA DEL ENTRENAMIENTO")

print("")

pd.DataFrame(history.history).plot(grid=True)

plt.show()

# Evaluar el modelo en el conjunto de entrenamiento

train\_loss, train\_accuracy = model.evaluate(training\_images, training\_labels)

print("Pérdida en el conjunto de entrenamiento:", train\_loss)

print("Precisión en el conjunto de entrenamiento:", train\_accuracy)

# Evaluar el modelo en el conjunto de prueba

test\_loss, test\_accuracy = model.evaluate(test\_images, test\_labels)

print("Pérdida en el conjunto de prueba:", test\_loss)

print("Precisión en el conjunto de prueba:", test\_accuracy)

# Predecir las etiquetas del conjunto de prueba

predictions = model.predict(test\_images)

predicted\_labels = np.argmax(predictions, axis=1)

# Matriz de confusión

print("MATRIZ DE CONFUSIÓN")

print("")

conf\_matrix = confusion\_matrix(test\_labels, predicted\_labels)

print("Matriz de confusión:\n", conf\_matrix) # Aquí se genera la matriz de confusión

# Reporte de clasificación

print( "REPORTE DE CLASIFICACIÓN")

print("")

class\_report = classification\_report(test\_labels, predicted\_labels)

print("Reporte de clasificación:\n", class\_report) # Aquí se genera el reporte de clasificación

# Cálculo del F1-score

print("CÁLCULO DEL F1-SCORE")

print("")

f1 = f1\_score(test\_labels, predicted\_labels, average='weighted')

print("F1-score:", f1) # Aquí se calcula el F1-score

# Graficar la matriz de confusión

print("GRAFICAR LA MATRIZ DE CONFUSIÓN")

print("")

plt.figure(figsize=(10, 7))

sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues")

plt.ylabel('Etiqueta verdadera')

plt.xlabel('Etiqueta predicha')

plt.title('Matriz de Confusión')

plt.show() # Aquí se grafica la matriz de confusión

# Histograma de la distribución del error

print("HISTOGRAMA DE LA DISTRIBUCIÓN DEL ERROR")

print("")

errors = test\_labels - predicted\_labels

plt.hist(errors, bins=30)

plt.xlabel('Error')

plt.ylabel('Frecuencia')

plt.title('Distribución del error')

plt.show() # Aquí se grafica la distribución del error en un histograma

Estructura de la Red Neuronal

Esta red tiene un total de 259274 parámetros que se entrenarán durante el proceso de aprendizaje. Cada capa tiene sus propios parámetros (pesos y biases) que contribuyen al total.

Layer (type) Output Shape Param #

=================================================================

flatten\_1 (Flatten) (None, 784) 0

dense\_3 (Dense) (None, 256) 200960

dense\_4 (Dense) (None, 128) 32896

dense\_5 (Dense) (None, 128) 16512

dense\_6 (Dense) (None, 64) 8256

dense\_7 (Dense) (None, 10) 650

=================================================================

Total params: 259274 (1012.79 KB)

Trainable params: 259274 (1012.79 KB)