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

```
Epoch 9/10
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0180 - accuracy: 0.9944
Epoch 10/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0142 - accuracy: 0.9958
```

CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 12

```
Epoch 9/12
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0181 - accuracy: 0.9944
Epoch 10/12
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0164 - accuracy: 0.9948
Epoch 11/12
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0130 - accuracy: 0.9960
Epoch 12/12
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0111 - accuracy: 0.9964
```

CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 14

```
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0115 - accuracy: 0.9963
Epoch 13/14
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0106 - accuracy: 0.9965
Epoch 14/14
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0089 - accuracy: 0.9972
```

CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 15

```
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0101 - accuracy: 0.9969
Epoch 14/15
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0081 - accuracy: 0.9973
Epoch 15/15
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0078 - accuracy: 0.9974
```

sobreajuste

CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 16

```
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0082 - accuracy: 0.9975
Epoch 15/16
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0091 - accuracy: 0.9972
Epoch 16/16
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0084 - accuracy: 0.9972
```

CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 17

```
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0081 - accuracy: 0.9973
Epoch 15/17
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0088 - accuracy: 0.9973
Epoch 16/17
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0074 - accuracy: 0.9977
Epoch 17/17
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0061 - accuracy: 0.9981
```

CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 18

```
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0066 - accuracy: 0.9979
Epoch 16/18
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0063 - accuracy: 0.9983
Epoch 17/18
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0069 - accuracy: 0.9977
Epoch 18/18
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0058 - accuracy: 0.9981
```

CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 19

```
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0085 - accuracy: 0.9972
Epoch 16/19
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0076 - accuracy: 0.9975
Epoch 17/19
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0080 - accuracy: 0.9973
Epoch 18/19
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0057 - accuracy: 0.9984
Epoch 19/19
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0063 - accuracy: 0.9981
```

CAMBIAMOS LOS EPOCHE QUE ESTABA EN 10 A 20

```
Epoch 15/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0061 - accuracy: 0.9983
Epoch 16/20
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0078 - accuracy: 0.9974
Epoch 17/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0075 - accuracy: 0.9977
Epoch 18/20
1875/1875 [==============================] - 6s 3ms/step - loss: 0.0057 - accuracy: 0.9982
Epoch 19/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0057 - accuracy: 0.9983
Epoch 20/20
1875/1875 [==============================] - 5s 3ms/step - loss: 0.0076 - accuracy: 0.9974
```

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)
```