

A3.2 Redes Neuronales

Luis Enrique Garcia Gallegos

Matricula: 649247

En esta actividad trabajarás con la base de datos de imágenes del MNIST, misma que se usó en el Taller de Redes Neuronales. Desarrolla los siguientes puntos en una *Jupyter Notebook*, tratando, dentro de lo posible, que cada punto se trabaje en una celda distinta. Los comentarios en el código siempre son bienvenidos, de preferencia, aprovecha el *markdown* para generar cuadros de descripción que ayuden al lector a comprender el trabajo realizado.

1. Entrena un modelo de redes neuronales para clasificar los dígitos de **0** al **9**, y muestra mediante gráficas el comportamiento del sistema a lo largo del entrenamiento. Especifica la exactitud del modelo tanto en entrenamiento como en validación.

```
In [ ]: import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import seaborn as sns
import cv2
from sklearn.metrics import confusion_matrix
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers
from matplotlib.ticker import MaxNLocator
(train_images, train_labels), (test_images, test_labels)=mnist.load_data()
modelo=Sequential()
modelo.add(Flatten(input_shape=(28, 28)))
modelo.add(Dense(256, activation='relu'))
modelo.add(Dense(128, activation='relu'))
modelo.add(Dense(64, activation='relu'))
modelo.add(Dense(10, activation='softmax'))
modelo.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
modelo.summary()
early_stop=EarlyStopping(monitor='val_accuracy', patience=15, restore_best_weights=True)
history=modelo.fit(train_images, train_labels, epochs=100, validation_split=0.4, batch_size=32, callbacks=[early_stop])
stopped_epoch=early_stop.stopped_epoch
best_epoch=stopped_epoch-early_stop.patience
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.axvline(x=best_epoch, color='r', linestyle='--', label='Early Stopping Epoch')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.gca().xaxis.set_major_locator(MaxNLocator(integer=True))
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.axvline(x=best_epoch, color='r', linestyle='--', label='Early Stopping Epoch')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.gca().xaxis.set_major_locator(MaxNLocator(integer=True))
plt.legend()
plt.tight_layout()
plt.show()
```

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|-------------------|--------------|---------|
| flatten (Flatten) | (None, 784) | 0 |
| dense (Dense) | (None, 256) | 200,960 |
| dense_1 (Dense) | (None, 128) | 32,896 |
| dense_2 (Dense) | (None, 64) | 8,256 |
| dense_3 (Dense) | (None, 10) | 650 |

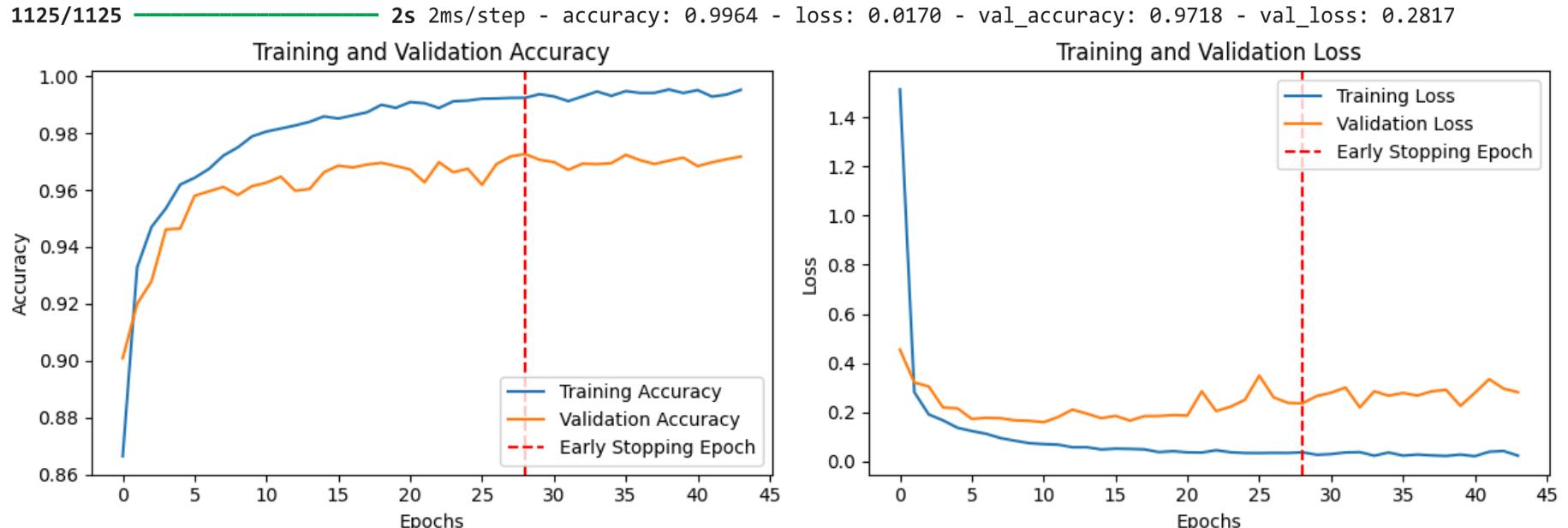
Total params: 242,762 (948.29 KB)

Trainable params: 242,762 (948.29 KB)

Non-trainable params: 0 (0.00 B)

Epoch 1/100
1125/1125 2s 2ms/step - accuracy: 0.8056 - loss: 3.9382 - val_accuracy: 0.9008 - val_loss: 0.4548
Epoch 2/100
1125/1125 2s 2ms/step - accuracy: 0.9296 - loss: 0.2977 - val_accuracy: 0.9200 - val_loss: 0.3212
Epoch 3/100
1125/1125 2s 2ms/step - accuracy: 0.9461 - loss: 0.1901 - val_accuracy: 0.9279 - val_loss: 0.3052
Epoch 4/100
1125/1125 2s 2ms/step - accuracy: 0.9531 - loss: 0.1667 - val_accuracy: 0.9461 - val_loss: 0.2195
Epoch 5/100
1125/1125 2s 2ms/step - accuracy: 0.9621 - loss: 0.1363 - val_accuracy: 0.9465 - val_loss: 0.2164
Epoch 6/100
1125/1125 2s 2ms/step - accuracy: 0.9674 - loss: 0.1162 - val_accuracy: 0.9580 - val_loss: 0.1733
Epoch 7/100
1125/1125 2s 2ms/step - accuracy: 0.9682 - loss: 0.1085 - val_accuracy: 0.9595 - val_loss: 0.1777
Epoch 8/100
1125/1125 2s 2ms/step - accuracy: 0.9750 - loss: 0.0855 - val_accuracy: 0.9611 - val_loss: 0.1761
Epoch 9/100
1125/1125 2s 2ms/step - accuracy: 0.9754 - loss: 0.0831 - val_accuracy: 0.9582 - val_loss: 0.1673
Epoch 10/100
1125/1125 2s 2ms/step - accuracy: 0.9784 - loss: 0.0728 - val_accuracy: 0.9614 - val_loss: 0.1654
Epoch 11/100
1125/1125 2s 2ms/step - accuracy: 0.9812 - loss: 0.0661 - val_accuracy: 0.9626 - val_loss: 0.1598
Epoch 12/100
1125/1125 2s 2ms/step - accuracy: 0.9843 - loss: 0.0562 - val_accuracy: 0.9647 - val_loss: 0.1807
Epoch 13/100
1125/1125 2s 2ms/step - accuracy: 0.9829 - loss: 0.0580 - val_accuracy: 0.9597 - val_loss: 0.2112
Epoch 14/100
1125/1125 2s 2ms/step - accuracy: 0.9842 - loss: 0.0534 - val_accuracy: 0.9604 - val_loss: 0.1946
Epoch 15/100
1125/1125 2s 2ms/step - accuracy: 0.9870 - loss: 0.0462 - val_accuracy: 0.9663 - val_loss: 0.1763
Epoch 16/100
1125/1125 2s 2ms/step - accuracy: 0.9854 - loss: 0.0497 - val_accuracy: 0.9686 - val_loss: 0.1855
Epoch 17/100
1125/1125 2s 2ms/step - accuracy: 0.9870 - loss: 0.0488 - val_accuracy: 0.9680 - val_loss: 0.1662
Epoch 18/100
1125/1125 2s 2ms/step - accuracy: 0.9871 - loss: 0.0490 - val_accuracy: 0.9690 - val_loss: 0.1844
Epoch 19/100
1125/1125 2s 2ms/step - accuracy: 0.9919 - loss: 0.0321 - val_accuracy: 0.9696 - val_loss: 0.1850
Epoch 20/100
1125/1125 2s 2ms/step - accuracy: 0.9902 - loss: 0.0339 - val_accuracy: 0.9685 - val_loss: 0.1887
Epoch 21/100
1125/1125 2s 2ms/step - accuracy: 0.9923 - loss: 0.0306 - val_accuracy: 0.9672 - val_loss: 0.1872
Epoch 22/100
1125/1125 2s 2ms/step - accuracy: 0.9924 - loss: 0.0296 - val_accuracy: 0.9627 - val_loss: 0.2853
Epoch 23/100
1125/1125 2s 2ms/step - accuracy: 0.9894 - loss: 0.0431 - val_accuracy: 0.9698 - val_loss: 0.2050
Epoch 24/100
1125/1125 2s 2ms/step - accuracy: 0.9927 - loss: 0.0289 - val_accuracy: 0.9663 - val_loss: 0.2226
Epoch 25/100
1125/1125 2s 2ms/step - accuracy: 0.9917 - loss: 0.0338 - val_accuracy: 0.9675 - val_loss: 0.2507
Epoch 26/100
1125/1125 2s 2ms/step - accuracy: 0.9912 - loss: 0.0368 - val_accuracy: 0.9618 - val_loss: 0.3492
Epoch 27/100
1125/1125 2s 2ms/step - accuracy: 0.9911 - loss: 0.0410 - val_accuracy: 0.9691 - val_loss: 0.2608
Epoch 28/100
1125/1125 2s 2ms/step - accuracy: 0.9925 - loss: 0.0367 - val_accuracy: 0.9718 - val_loss: 0.2384
Epoch 29/100
1125/1125 2s 2ms/step - accuracy: 0.9921 - loss: 0.0421 - val_accuracy: 0.9727 - val_loss: 0.2366
Epoch 30/100
1125/1125 2s 2ms/step - accuracy: 0.9940 - loss: 0.0250 - val_accuracy: 0.9707 - val_loss: 0.2661
Epoch 31/100
1125/1125 2s 2ms/step - accuracy: 0.9933 - loss: 0.0315 - val_accuracy: 0.9698 - val_loss: 0.2797
Epoch 32/100
1125/1125 2s 2ms/step - accuracy: 0.9903 - loss: 0.0430 - val_accuracy: 0.9671 - val_loss: 0.3009
Epoch 33/100
1125/1125 2s 1ms/step - accuracy: 0.9942 - loss: 0.0288 - val_accuracy: 0.9693 - val_loss: 0.2199
Epoch 34/100
1125/1125 2s 2ms/step - accuracy: 0.9949 - loss: 0.0220 - val_accuracy: 0.9691 - val_loss: 0.2854
Epoch 35/100
1125/1125 2s 2ms/step - accuracy: 0.9935 - loss: 0.0335 - val_accuracy: 0.9695 - val_loss: 0.2676
Epoch 36/100
1125/1125 2s 2ms/step - accuracy: 0.9941 - loss: 0.0293 - val_accuracy: 0.9724 - val_loss: 0.2787
Epoch 37/100
1125/1125 2s 1ms/step - accuracy: 0.9941 - loss: 0.0306 - val_accuracy: 0.9705 - val_loss: 0.2680
Epoch 38/100
1125/1125 2s 2ms/step - accuracy: 0.9952 - loss: 0.0206 - val_accuracy: 0.9692 - val_loss: 0.2855
Epoch 39/100
1125/1125 2s 2ms/step - accuracy: 0.9952 - loss: 0.0247 - val_accuracy: 0.9703 - val_loss: 0.2913
Epoch 40/100
1125/1125 2s 2ms/step - accuracy: 0.9965 - loss: 0.0155 - val_accuracy: 0.9714 - val_loss: 0.2261
Epoch 41/100
1125/1125 2s 2ms/step - accuracy: 0.9951 - loss: 0.0211 - val_accuracy: 0.9684 - val_loss: 0.2791
Epoch 42/100
1125/1125 2s 2ms/step - accuracy: 0.9934 - loss: 0.0345 - val_accuracy: 0.9697 - val_loss: 0.3346
Epoch 43/100
1125/1125 2s 2ms/step - accuracy: 0.9937 - loss: 0.0389 - val_accuracy: 0.9708 - val_loss: 0.2962

Epoch 44/100
1125/1125 2s 2ms/step - accuracy: 0.9964 - loss: 0.0170 - val_accuracy: 0.9718 - val_loss: 0.2817



Este es un modelo de redes neuronales muy simple realizado durante el taller de redes neuronales, como se ve en el histograma vemos que hay dejá de haber diferencias significativas alrededor del modelo 28, además de que entre el entrenamiento y validación notamos que hay diferencias notables.

2. Evalúa la exactitud del modelo en el conjunto de prueba generado al cargar la base de datos y compárala con la exactitud de validación del modelo. Realiza un comentario sobre los resultados de dicha comparación.

```
In [2]: test_loss, test_accuracy=modelo.evaluate(test_images, test_labels)
print(f'Exactitud en el conjunto de prueba: {test_accuracy:.4f}')
best_val_accuracy=max(history.history['val_accuracy'])
print(f'Máxima exactitud en validación: {best_val_accuracy:.4f}'')
```

313/313 0s 693us/step - accuracy: 0.9708 - loss: 0.2768
Exactitud en el conjunto de prueba: 0.9762
Máxima exactitud en validación: 0.9727

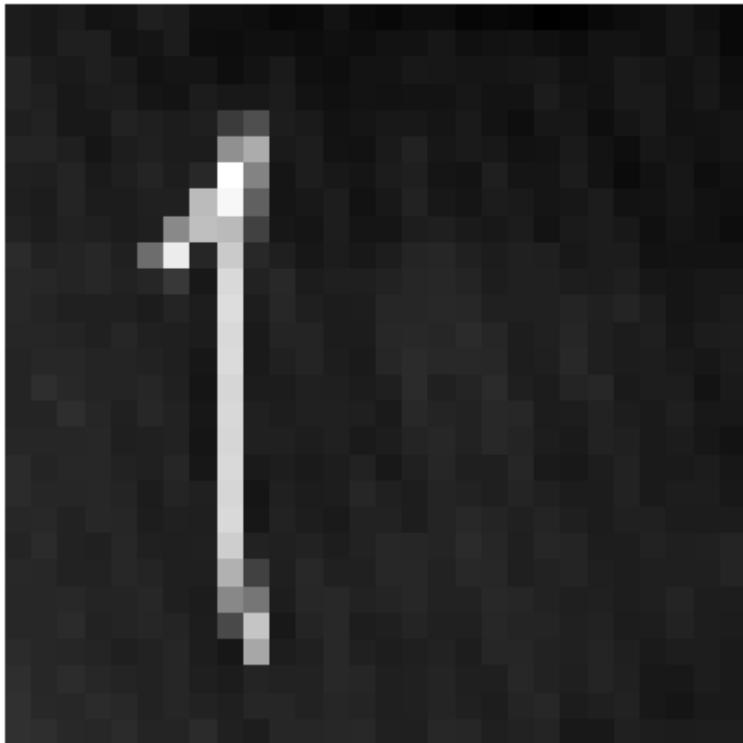
Este código fue proporcionado por ChatGpt y algo importante a destacar es que cuando probamos los datos de entrenamiento vemos que los accuracy son bastante similares aunque un poco diferente que el de entrenamiento.

3. Genera **50 imágenes, 5** para cada dígito. Preprocesa las imágenes para que puedan ser evaluadas por el modelo, de forma similar a como se realizó en el taller.

```
In [3]: true_labels={1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 2, 7: 2, 8: 2, 9: 2, 10: 2, 11: 3, 12: 3, 13: 3, 14: 3, 15: 3, 16: 4, 17: 4, 18: y_true=[true_labels[i] for i in range(1, 51)]
y_pred=[]
correct_count=0
total_count=0
def convert_to_mnist_format(image_path):
    img=cv2.imread(image_path)
    gray=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray=255-gray
    resized=cv2.resize(gray, (28, 28), interpolation=cv2.INTER_AREA)
    final_image=resized.reshape(1, 28, 28, 1)
    return final_image
for i in range(1, 51):
    image_path=f'imagen{i}.jpg'
    final_image=convert_to_mnist_format(image_path)
    prediction=modelo.predict(final_image)
    predicted_label=np.argmax(prediction, axis=1)[0]
    y_pred.append(predicted_label)
    true_label=true_labels[i]
    if predicted_label==true_label:
        correct_count+=1
    total_count+=1
    print(f'Predicción: {predicted_label}, Etiqueta verdadera: {true_label}')
    plt.imshow(final_image[0].reshape(28, 28), cmap='gray')
    plt.title(f'Predicción: {predicted_label}, Etiqueta verdadera: {true_label}')
    plt.axis('off')
    plt.show()
```

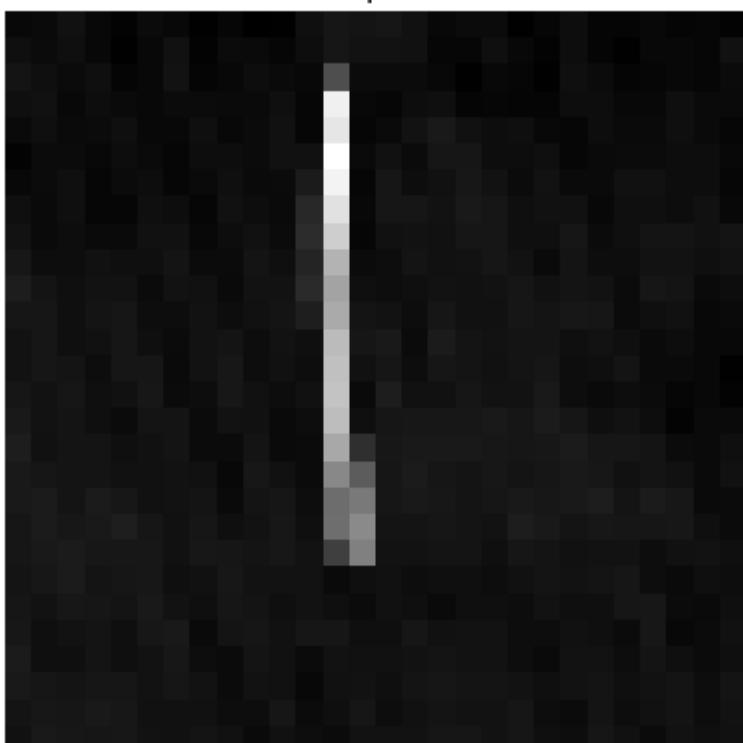
1/1 0s 41ms/step
Predicción: 8, Etiqueta verdadera: 1

Predicción: 8, Etiqueta verdadera: 1



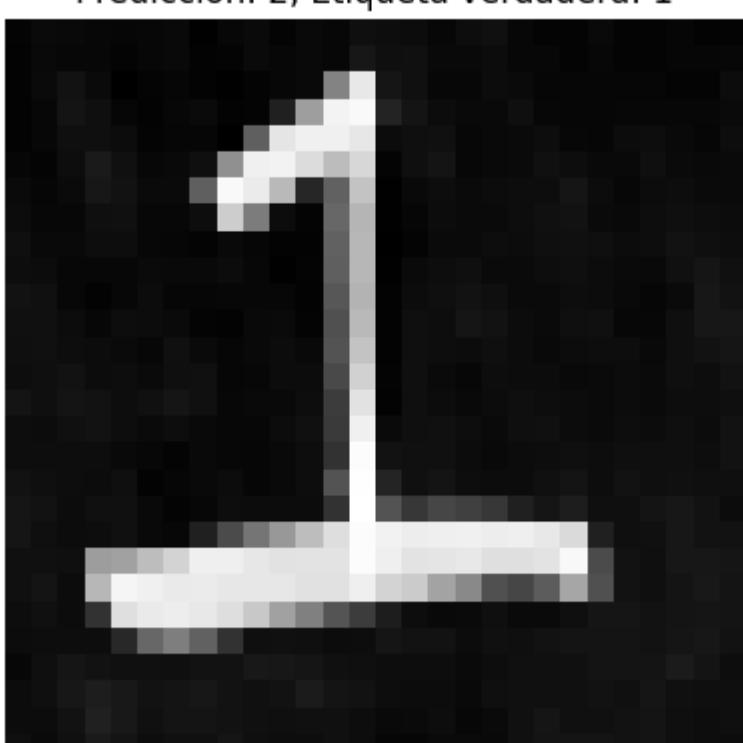
1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 1

Predicción: 8, Etiqueta verdadera: 1



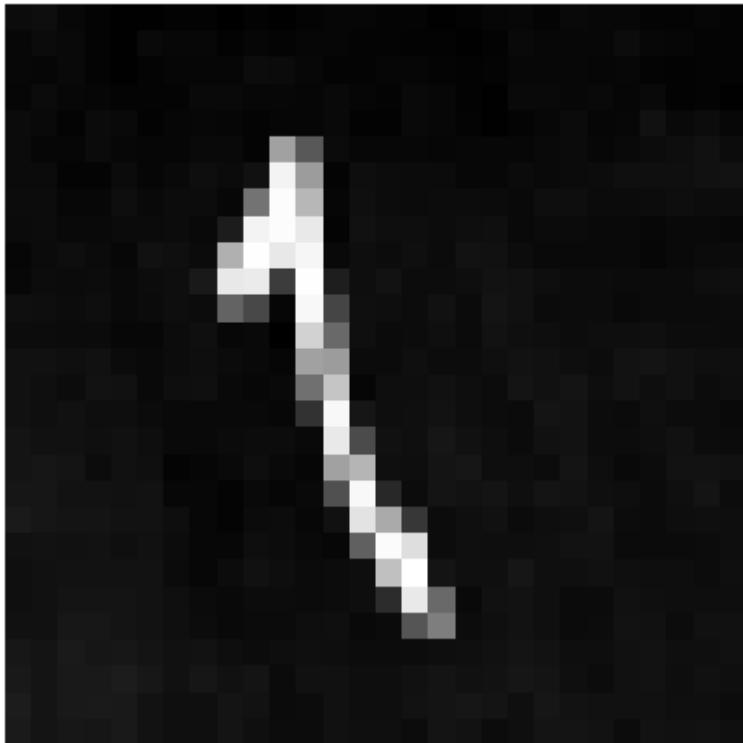
1/1 ————— 0s 30ms/step
Predicción: 2, Etiqueta verdadera: 1

Predicción: 2, Etiqueta verdadera: 1



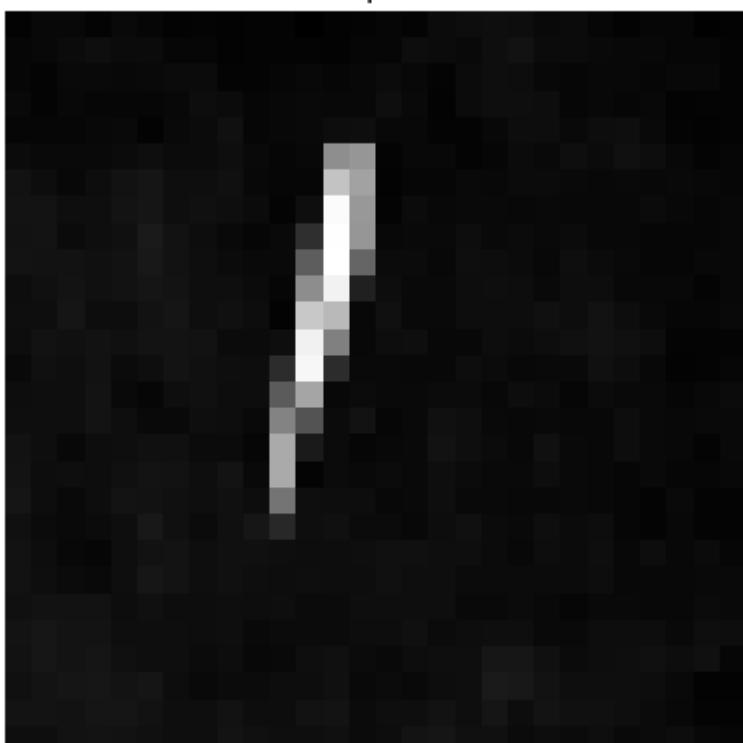
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 1

Predicción: 8, Etiqueta verdadera: 1



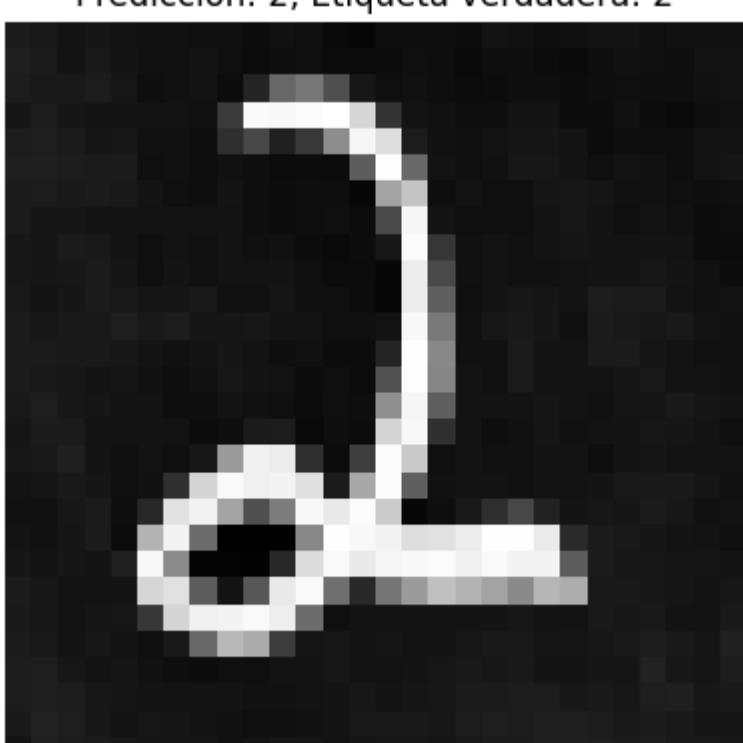
1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 1

Predicción: 8, Etiqueta verdadera: 1



1/1 ————— 0s 19ms/step
Predicción: 2, Etiqueta verdadera: 2

Predicción: 2, Etiqueta verdadera: 2



1/1 ————— 0s 21ms/step
Predicción: 2, Etiqueta verdadera: 2

Predicción: 2, Etiqueta verdadera: 2



1/1 ————— 0s 20ms/step
Predicción: 2, Etiqueta verdadera: 2

Predicción: 2, Etiqueta verdadera: 2



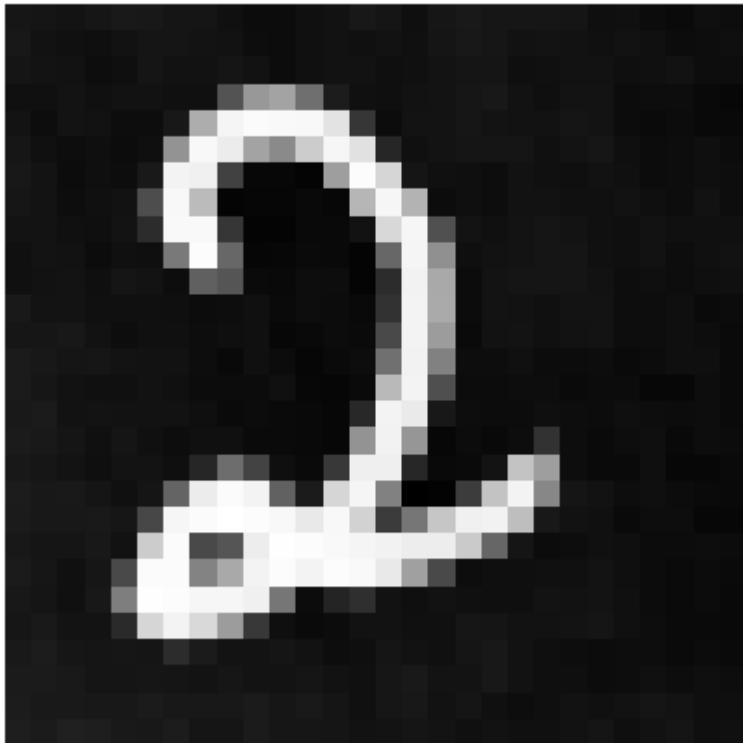
1/1 ————— 0s 20ms/step
Predicción: 9, Etiqueta verdadera: 2

Predicción: 9, Etiqueta verdadera: 2



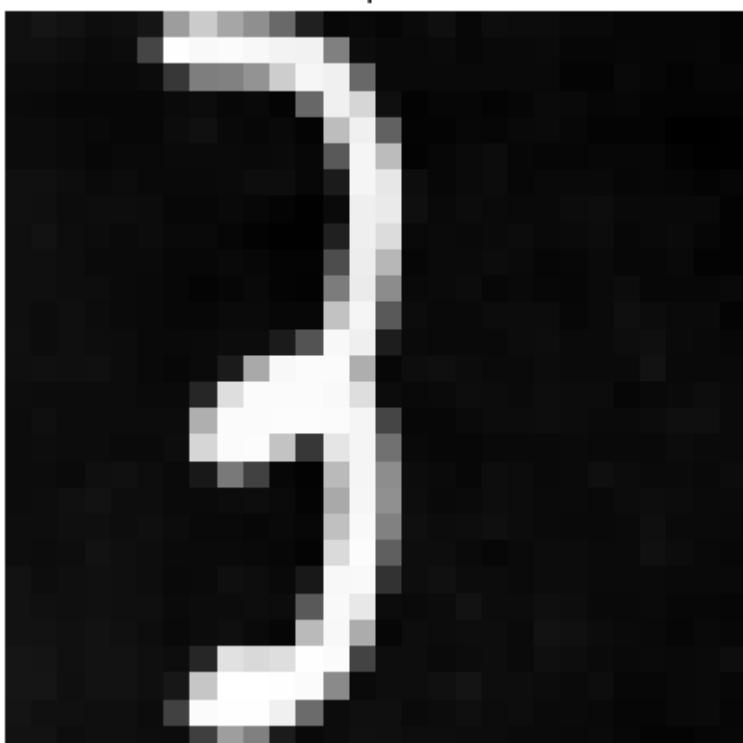
1/1 ————— 0s 22ms/step
Predicción: 3, Etiqueta verdadera: 2

Predicción: 3, Etiqueta verdadera: 2



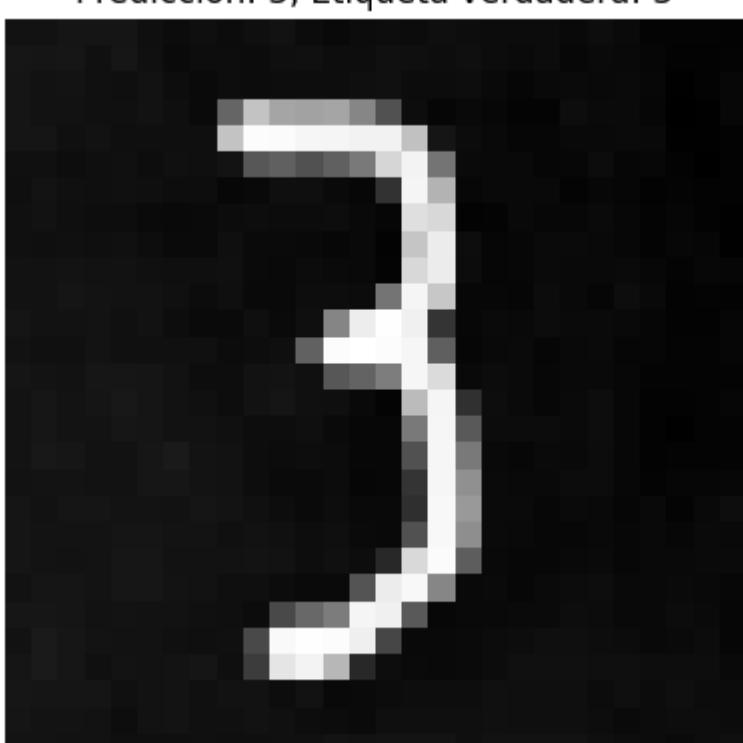
1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 3

Predicción: 8, Etiqueta verdadera: 3



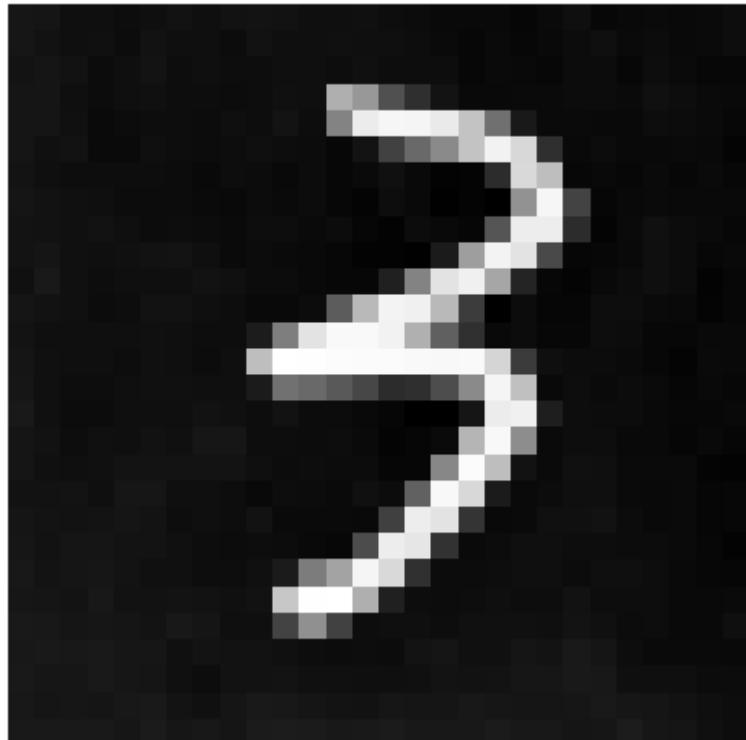
1/1 ————— 0s 21ms/step
Predicción: 3, Etiqueta verdadera: 3

Predicción: 3, Etiqueta verdadera: 3



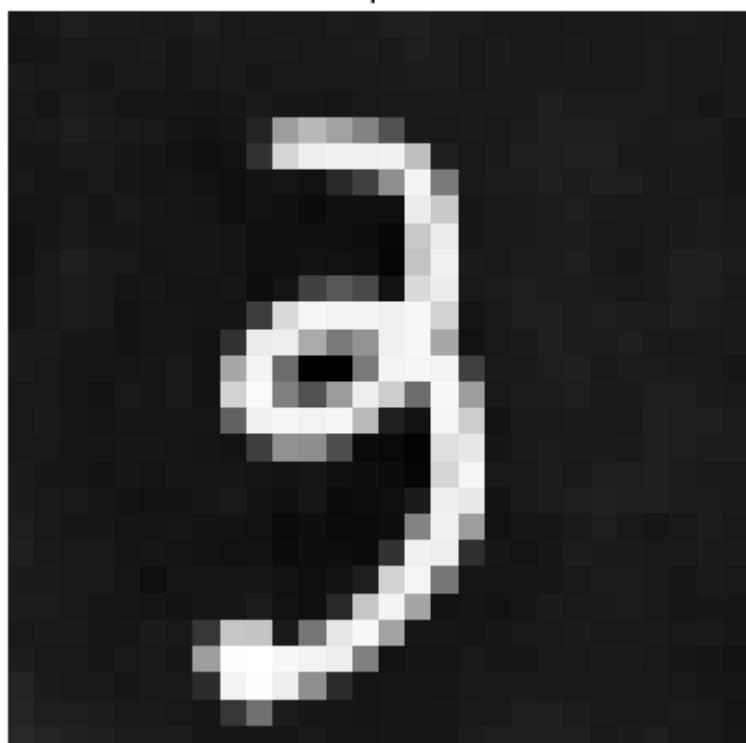
1/1 ————— 0s 20ms/step
Predicción: 3, Etiqueta verdadera: 3

Predicción: 3, Etiqueta verdadera: 3



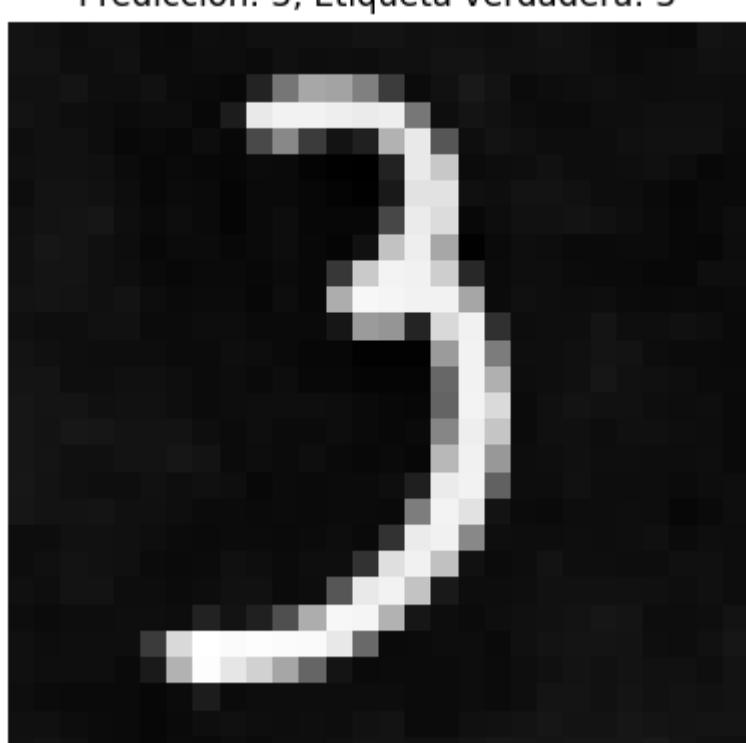
1/1 ————— 0s 19ms/step
Predicción: 3, Etiqueta verdadera: 3

Predicción: 3, Etiqueta verdadera: 3



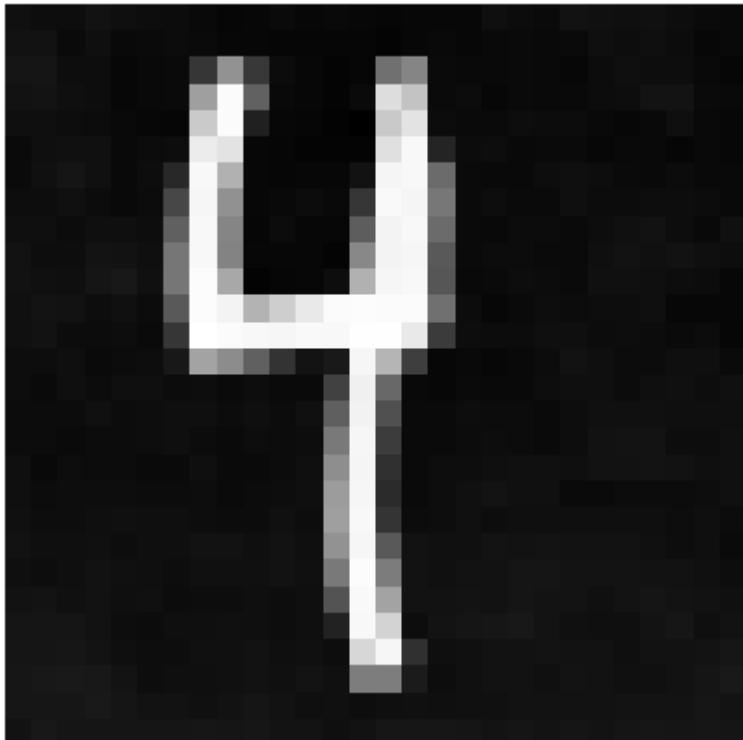
1/1 ————— 0s 20ms/step
Predicción: 3, Etiqueta verdadera: 3

Predicción: 3, Etiqueta verdadera: 3



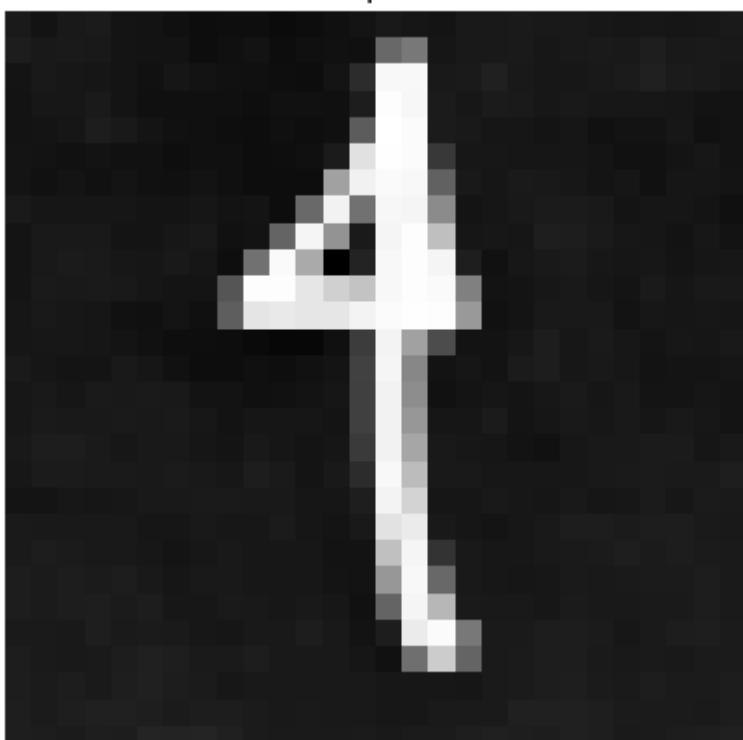
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 4

Predicción: 8, Etiqueta verdadera: 4



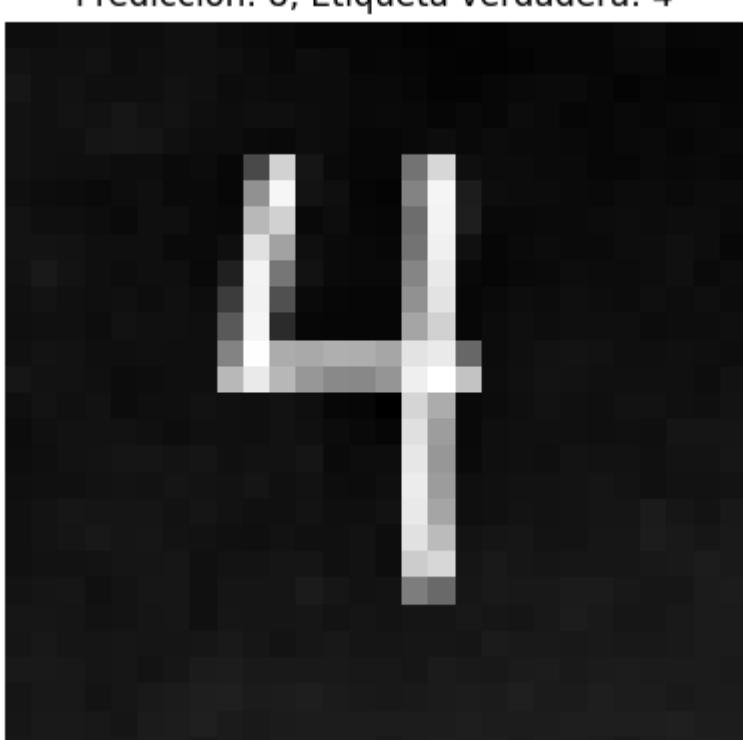
1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 4

Predicción: 8, Etiqueta verdadera: 4



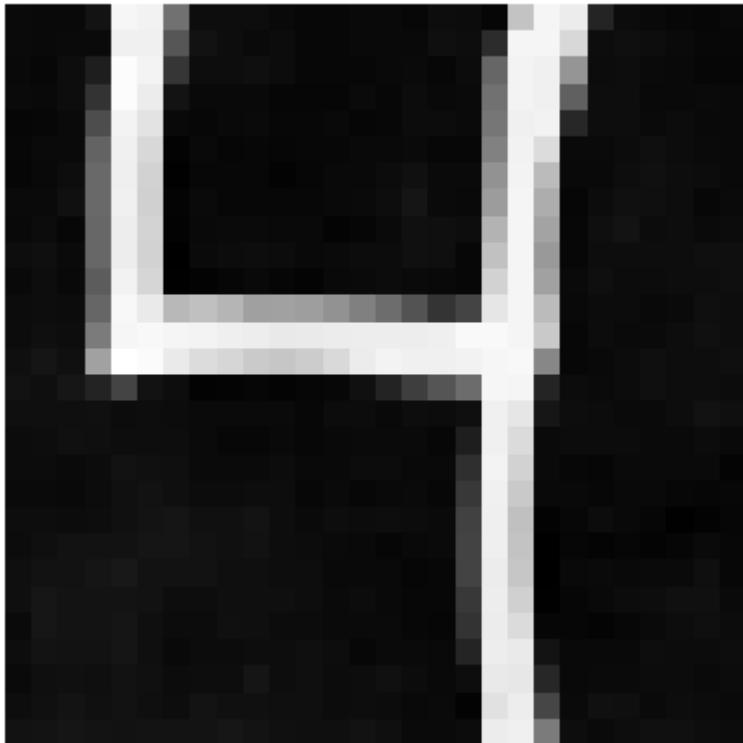
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 4

Predicción: 8, Etiqueta verdadera: 4



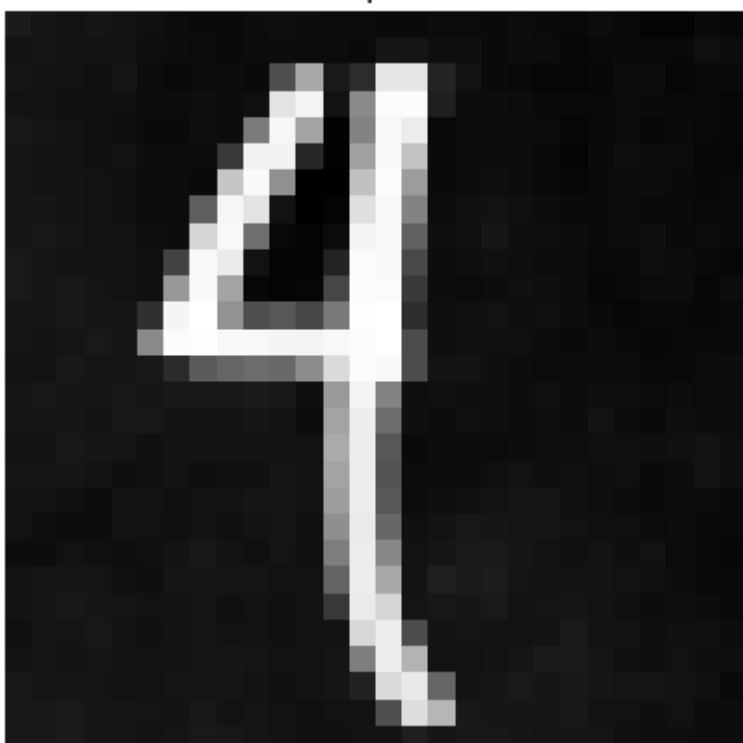
1/1 ————— 0s 21ms/step
Predicción: 8, Etiqueta verdadera: 4

Predicción: 8, Etiqueta verdadera: 4



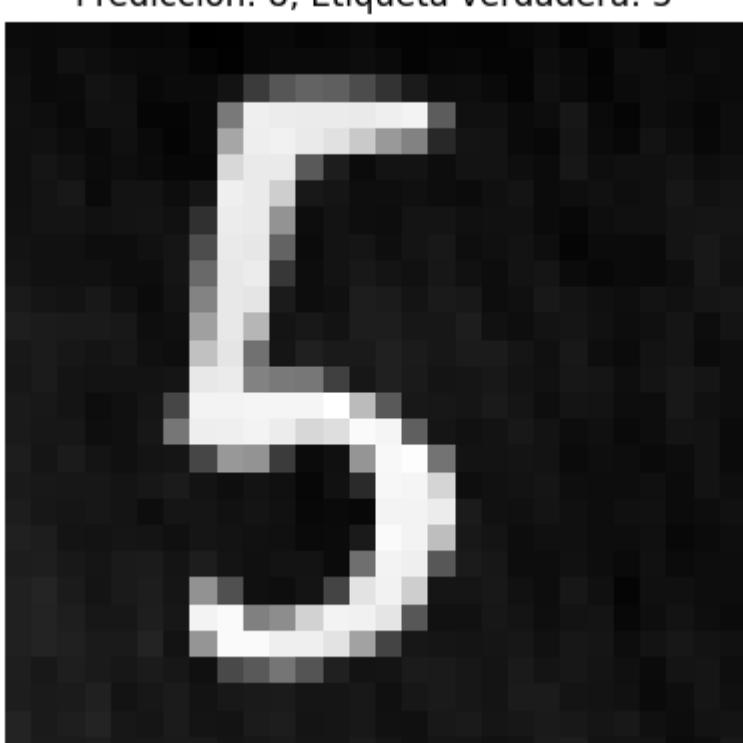
1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 4

Predicción: 8, Etiqueta verdadera: 4



1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 5

Predicción: 8, Etiqueta verdadera: 5



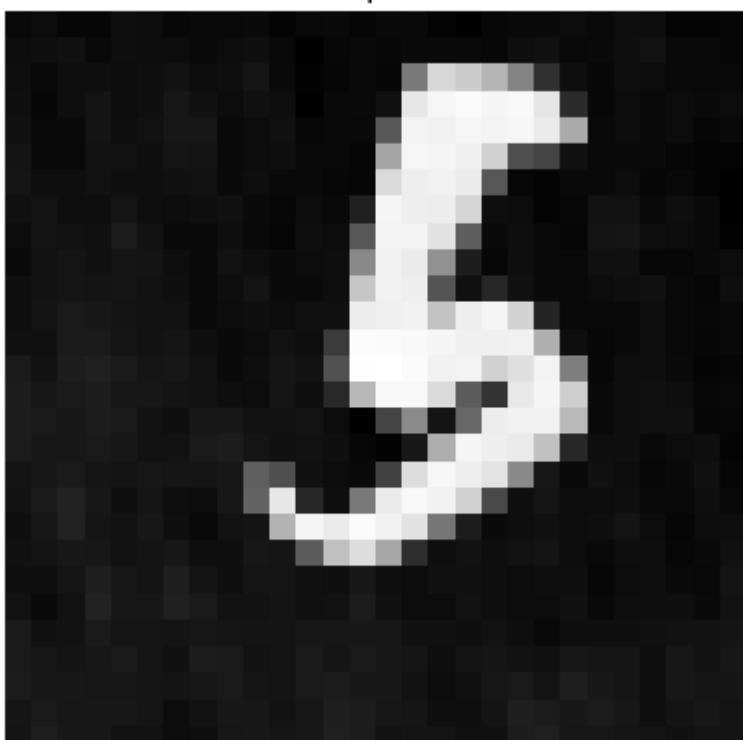
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 5

Predicción: 8, Etiqueta verdadera: 5



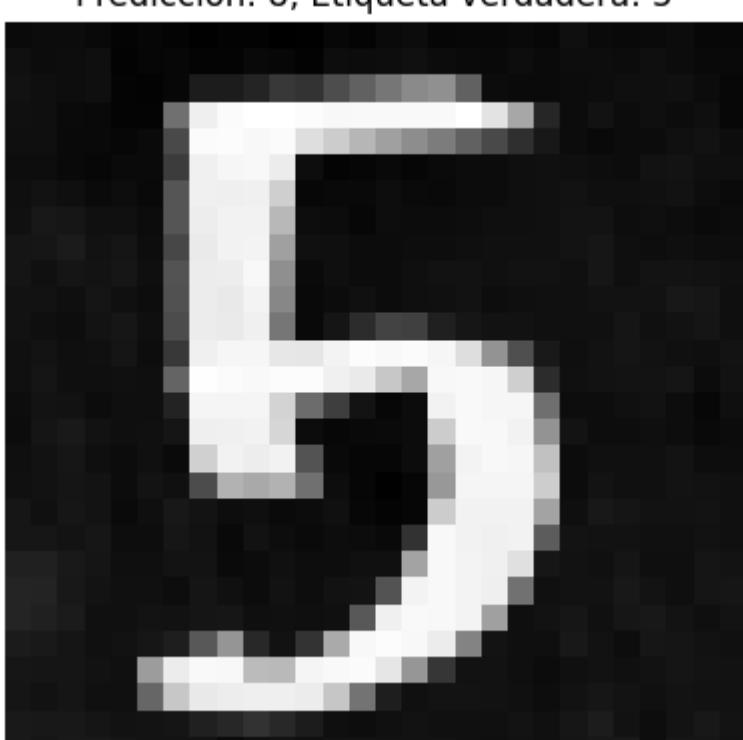
1/1 ————— 0s 21ms/step
Predicción: 3, Etiqueta verdadera: 5

Predicción: 3, Etiqueta verdadera: 5



1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 5

Predicción: 8, Etiqueta verdadera: 5



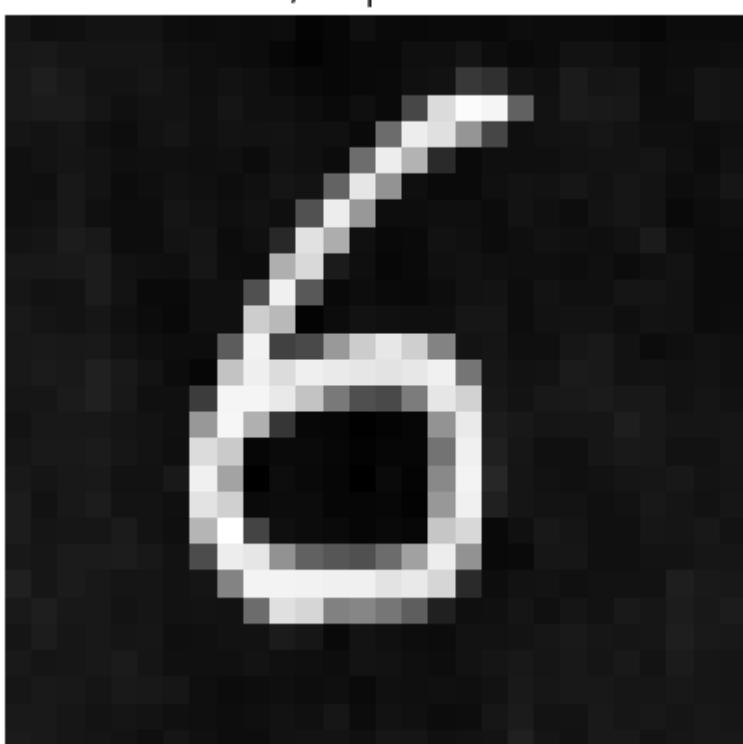
1/1 ————— 0s 19ms/step
Predicción: 5, Etiqueta verdadera: 5

Predicción: 5, Etiqueta verdadera: 5



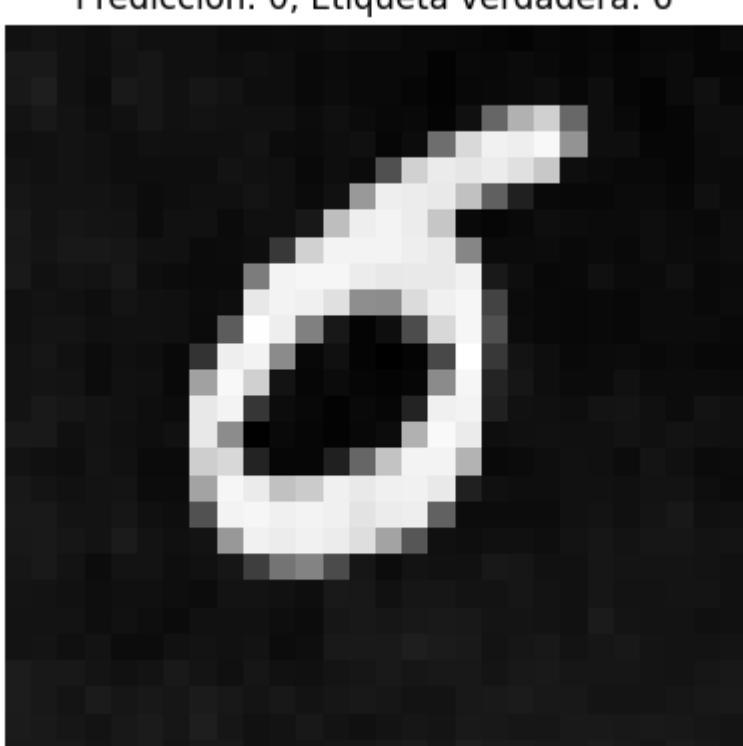
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 6

Predicción: 8, Etiqueta verdadera: 6



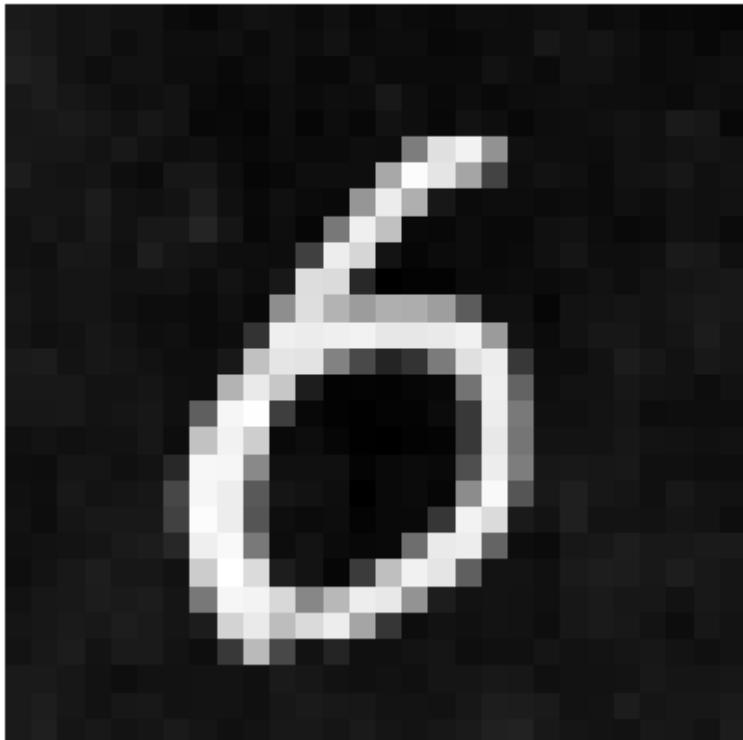
1/1 ————— 0s 20ms/step
Predicción: 0, Etiqueta verdadera: 6

Predicción: 0, Etiqueta verdadera: 6



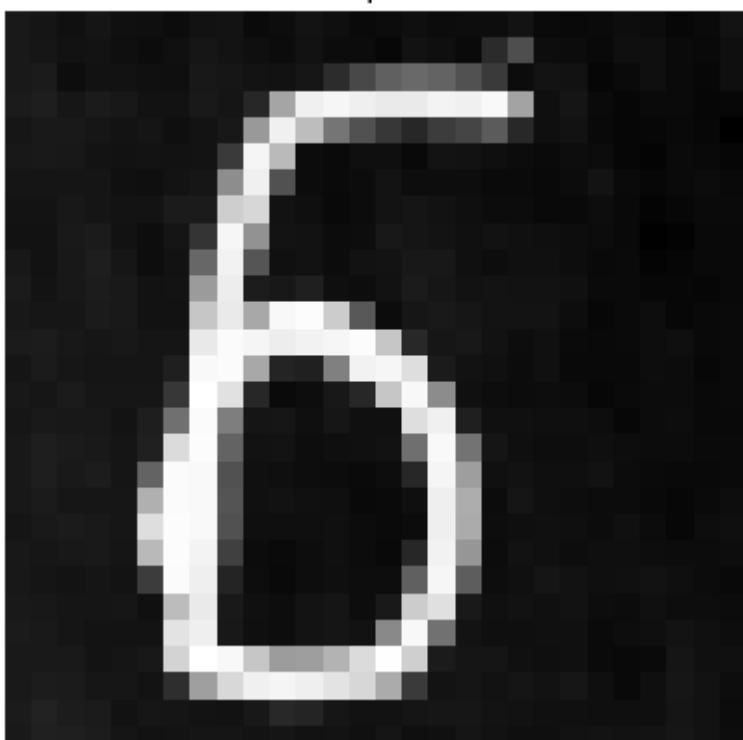
1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 6

Predicción: 8, Etiqueta verdadera: 6



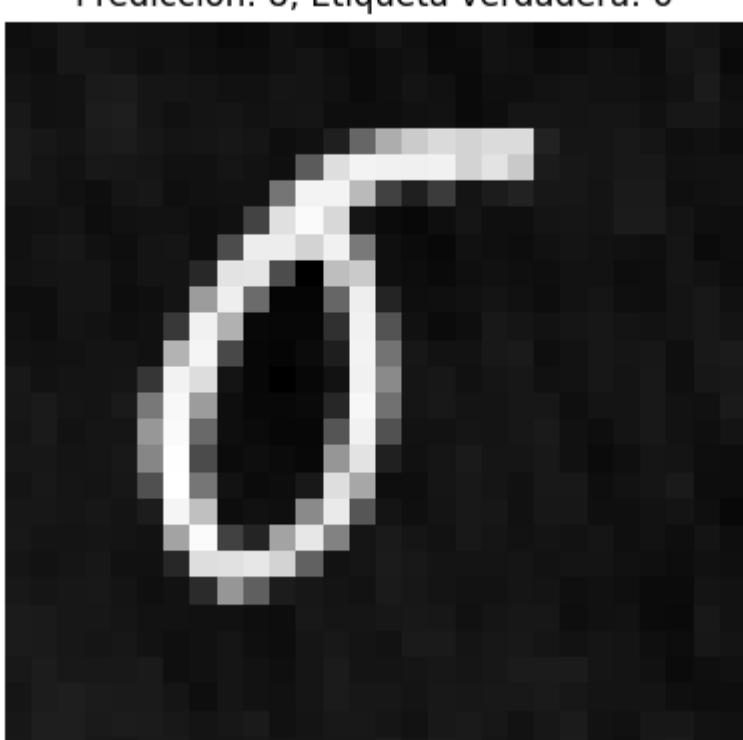
1/1 ————— 0s 19ms/step
Predicción: 3, Etiqueta verdadera: 6

Predicción: 3, Etiqueta verdadera: 6



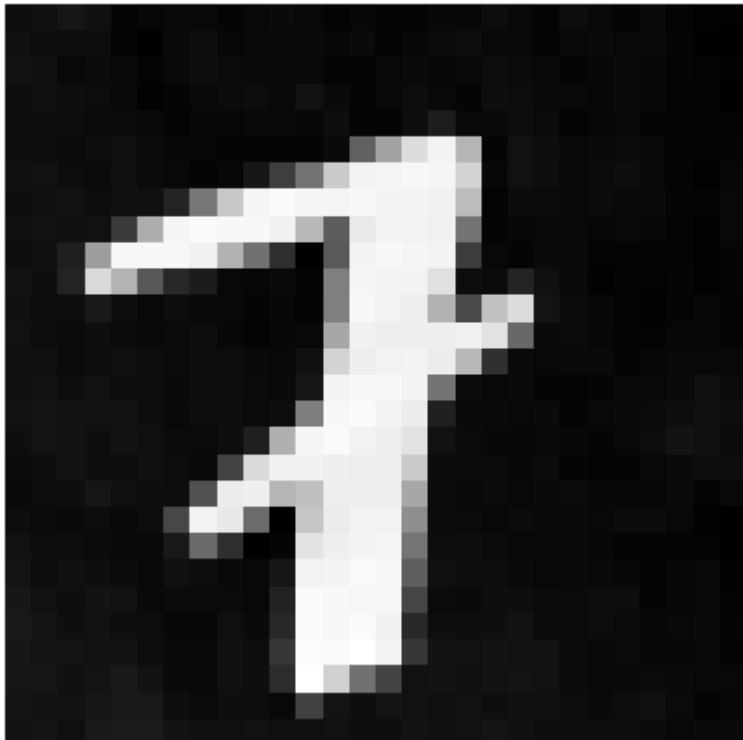
1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 6

Predicción: 8, Etiqueta verdadera: 6



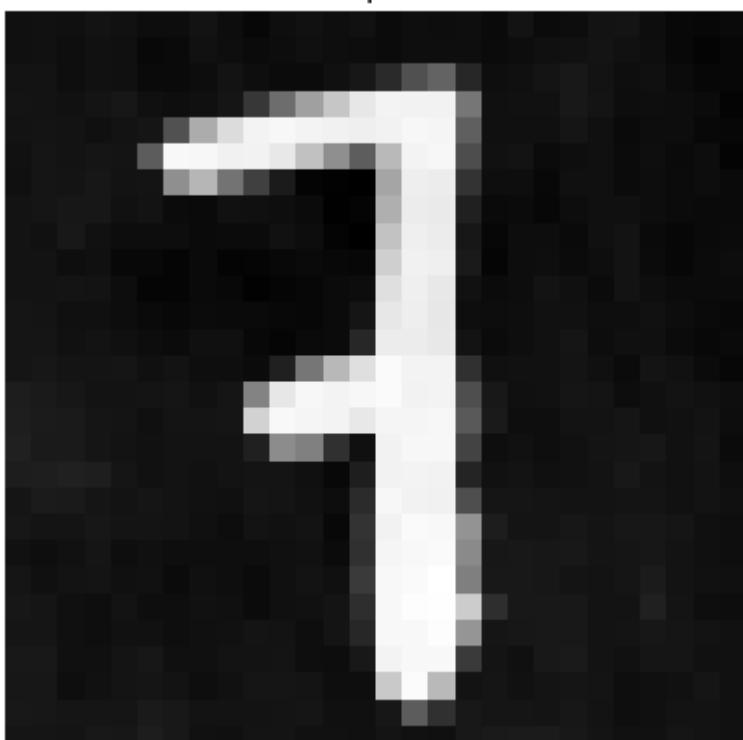
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 7

Predicción: 8, Etiqueta verdadera: 7



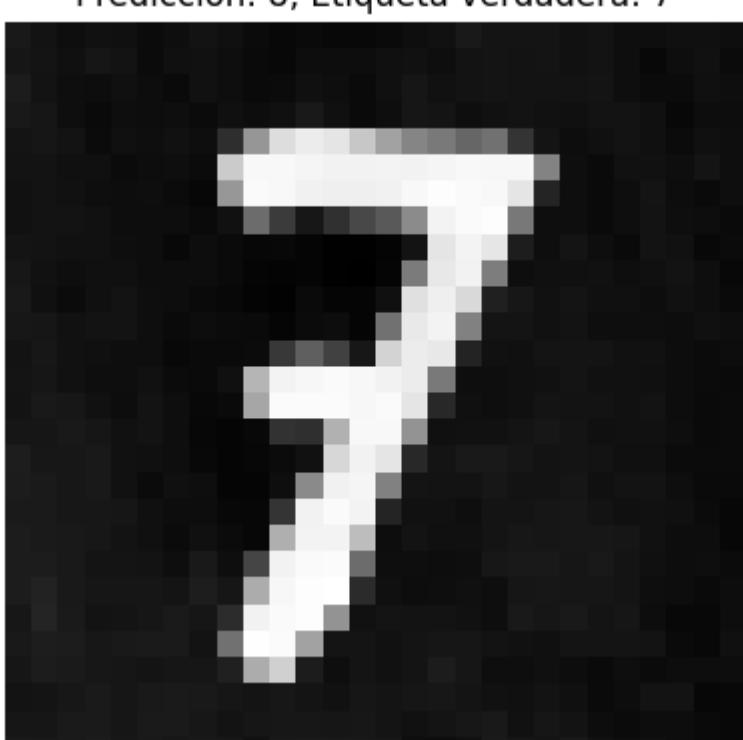
1/1 ————— 0s 20ms/step
Predicción: 7, Etiqueta verdadera: 7

Predicción: 7, Etiqueta verdadera: 7



1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 7

Predicción: 8, Etiqueta verdadera: 7



1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 7

Predicción: 8, Etiqueta verdadera: 7



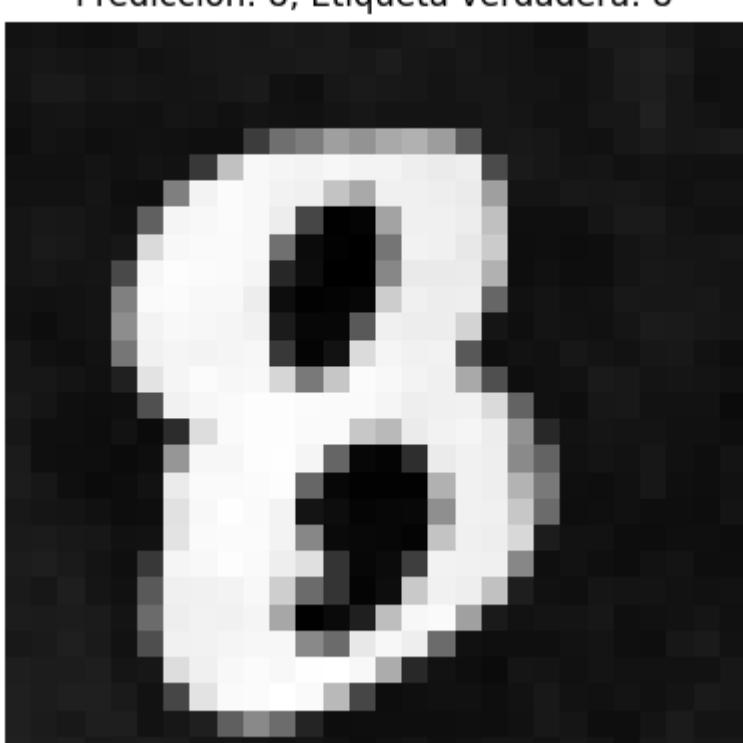
1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 7

Predicción: 8, Etiqueta verdadera: 7



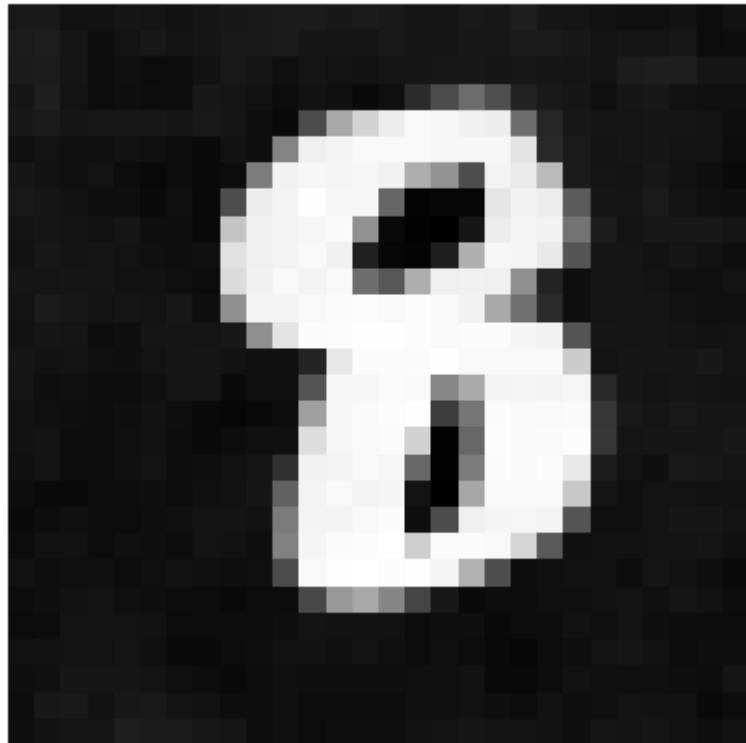
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 8

Predicción: 8, Etiqueta verdadera: 8



1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 8

Predicción: 8, Etiqueta verdadera: 8



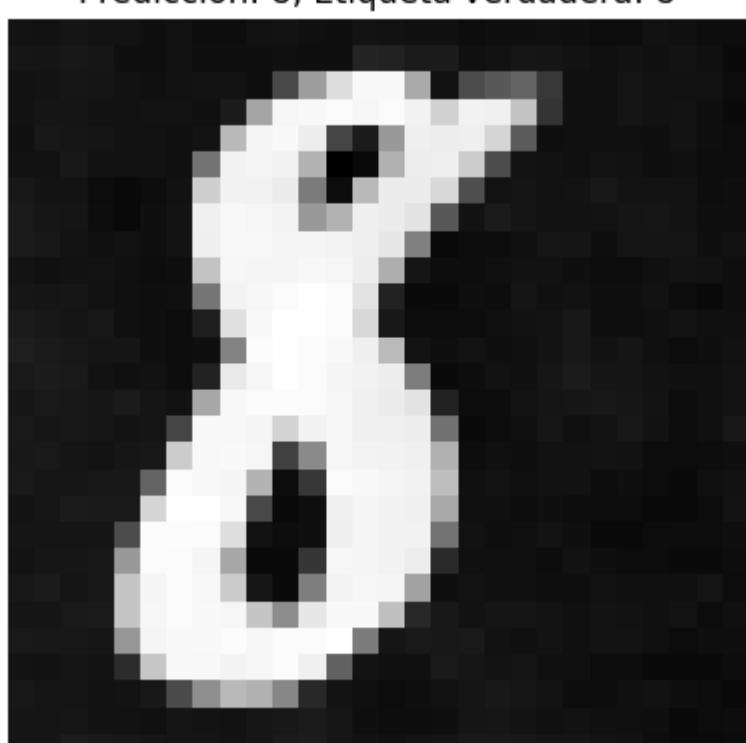
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 8

Predicción: 8, Etiqueta verdadera: 8



1/1 ————— 0s 21ms/step
Predicción: 8, Etiqueta verdadera: 8

Predicción: 8, Etiqueta verdadera: 8



1/1 ————— 0s 27ms/step
Predicción: 8, Etiqueta verdadera: 8

Predicción: 8, Etiqueta verdadera: 8



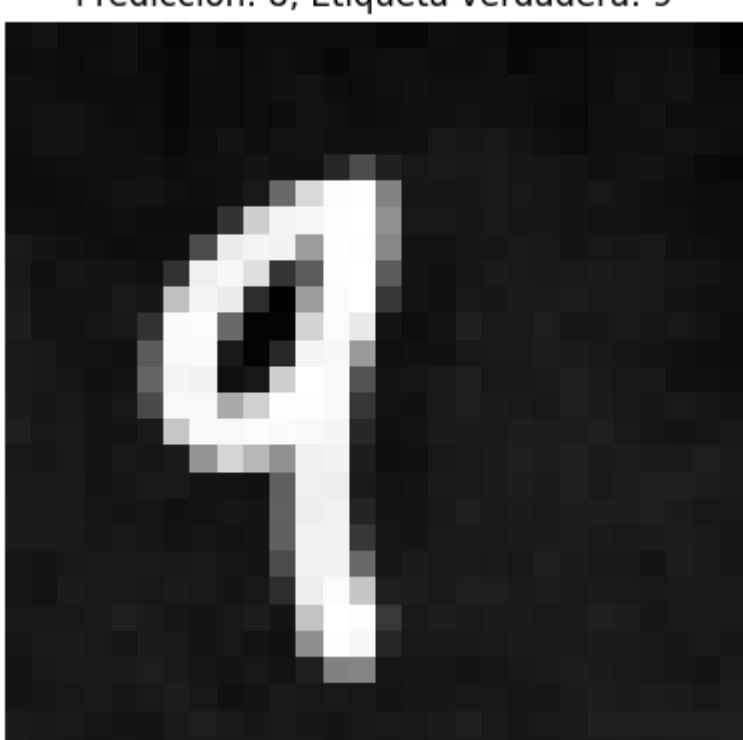
1/1 ————— 0s 18ms/step
Predicción: 8, Etiqueta verdadera: 9

Predicción: 8, Etiqueta verdadera: 9



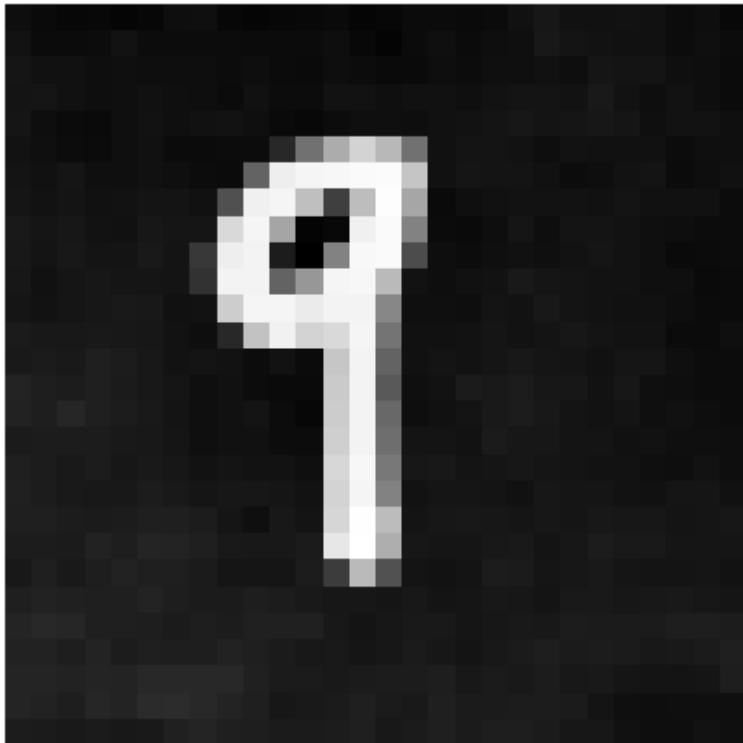
1/1 ————— 0s 18ms/step
Predicción: 8, Etiqueta verdadera: 9

Predicción: 8, Etiqueta verdadera: 9



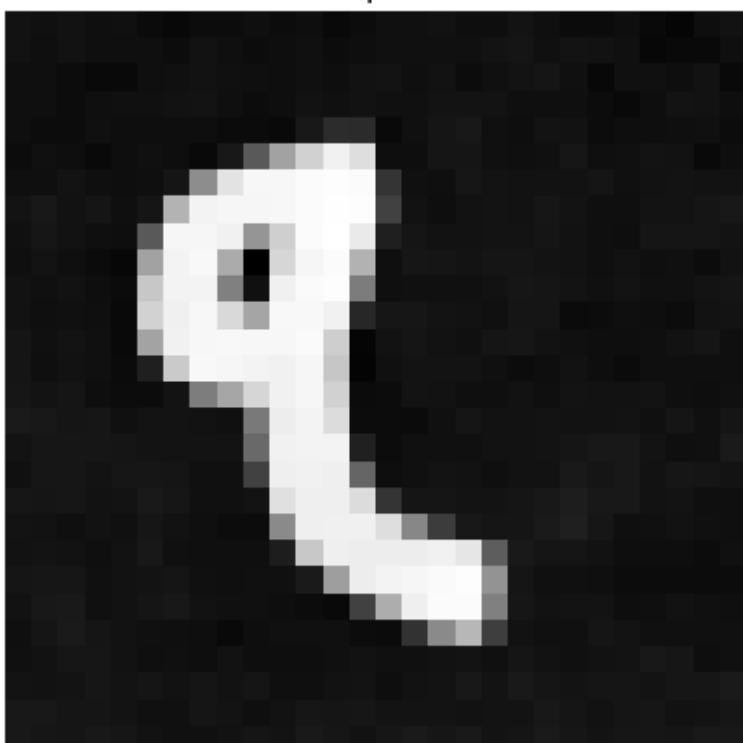
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 9

Predicción: 8, Etiqueta verdadera: 9



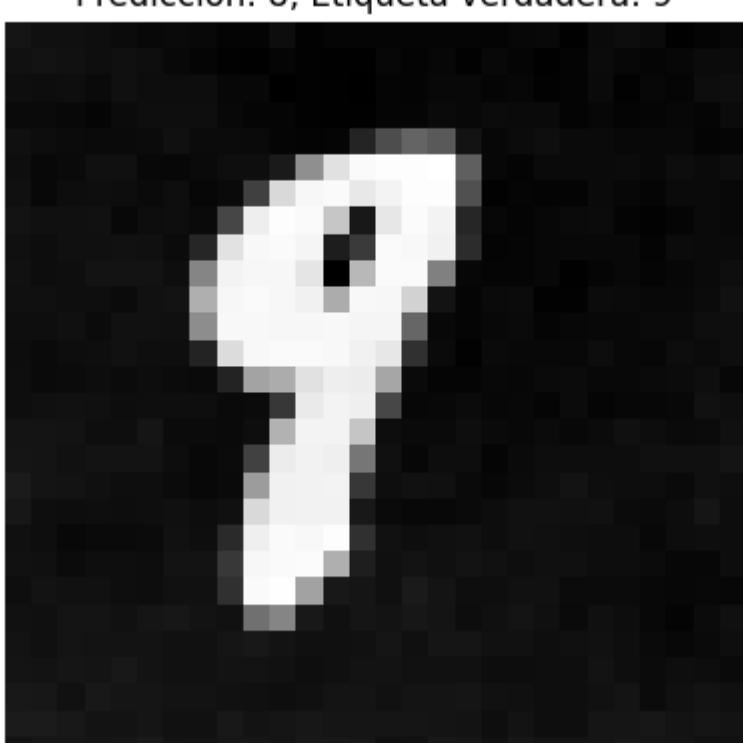
1/1 ————— 0s 20ms/step
Predicción: 0, Etiqueta verdadera: 9

Predicción: 0, Etiqueta verdadera: 9



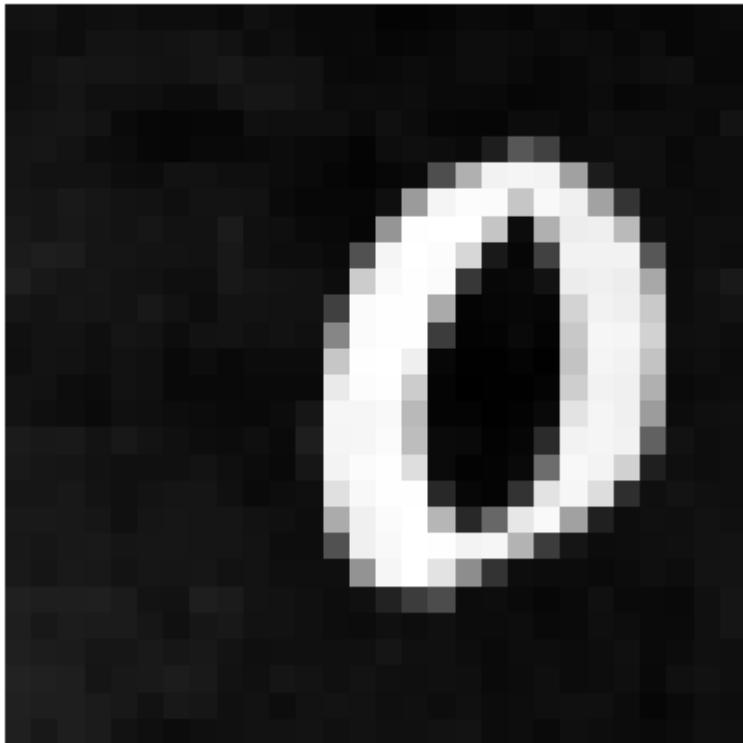
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 9

Predicción: 8, Etiqueta verdadera: 9



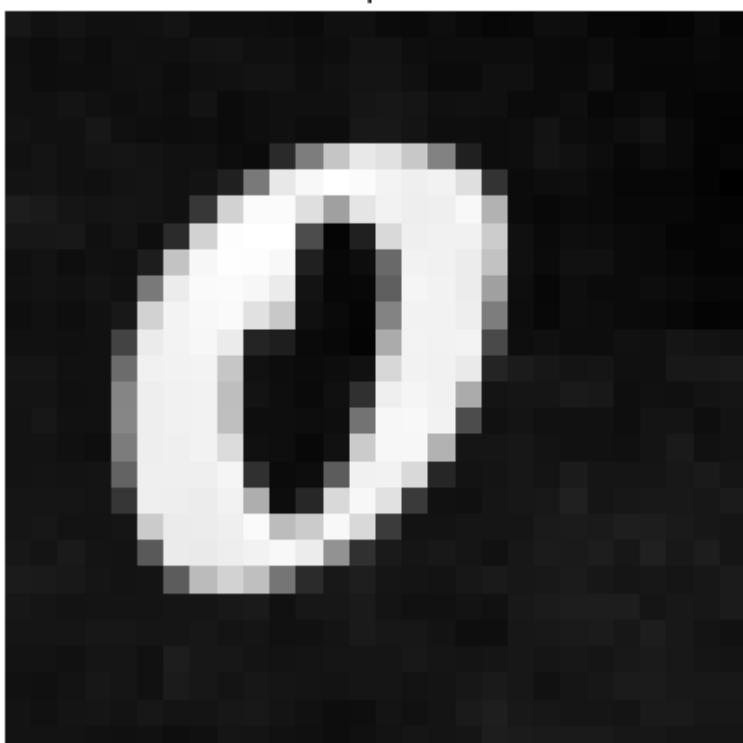
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 0

Predicción: 8, Etiqueta verdadera: 0



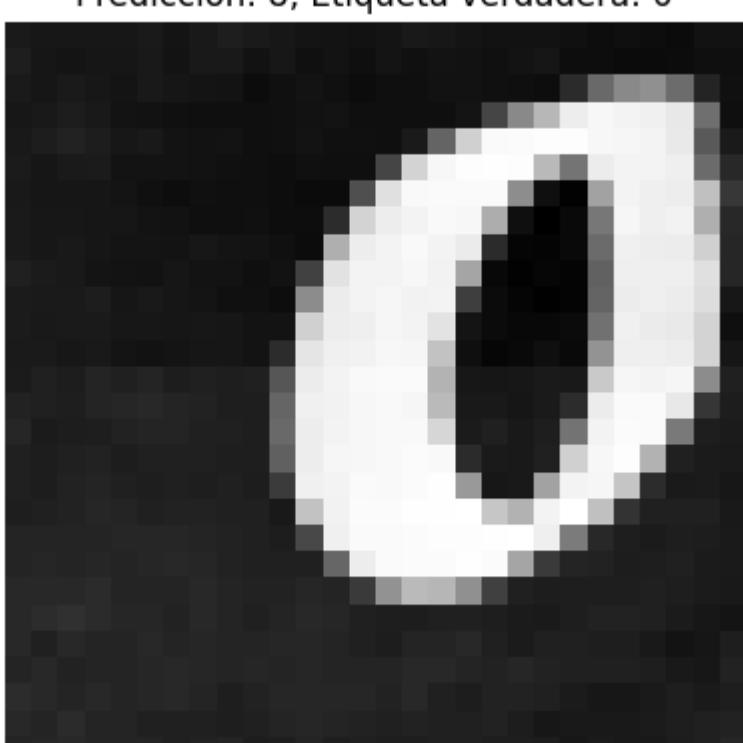
1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 0

Predicción: 8, Etiqueta verdadera: 0



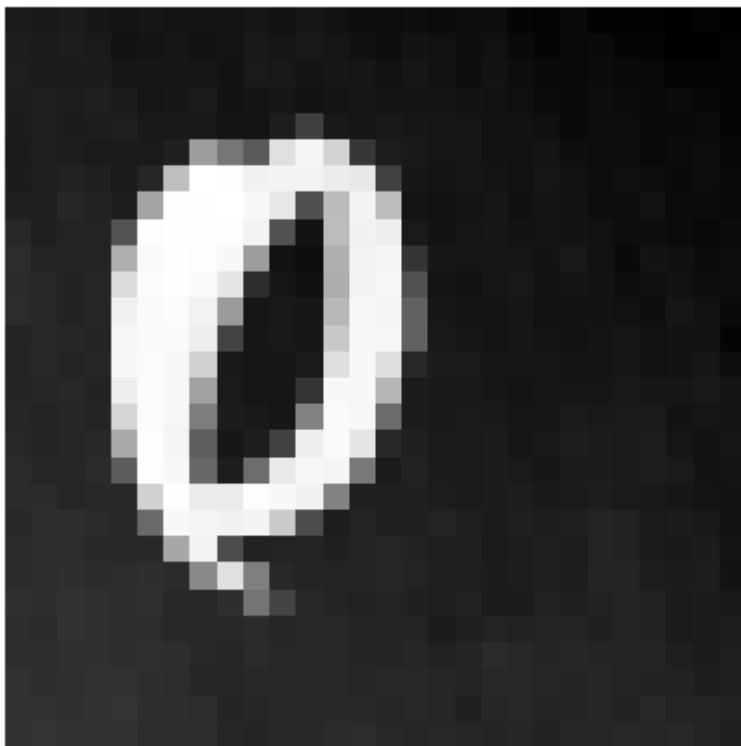
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 0

Predicción: 8, Etiqueta verdadera: 0



1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 0

Predicción: 8, Etiqueta verdadera: 0



1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 0

Predicción: 8, Etiqueta verdadera: 0

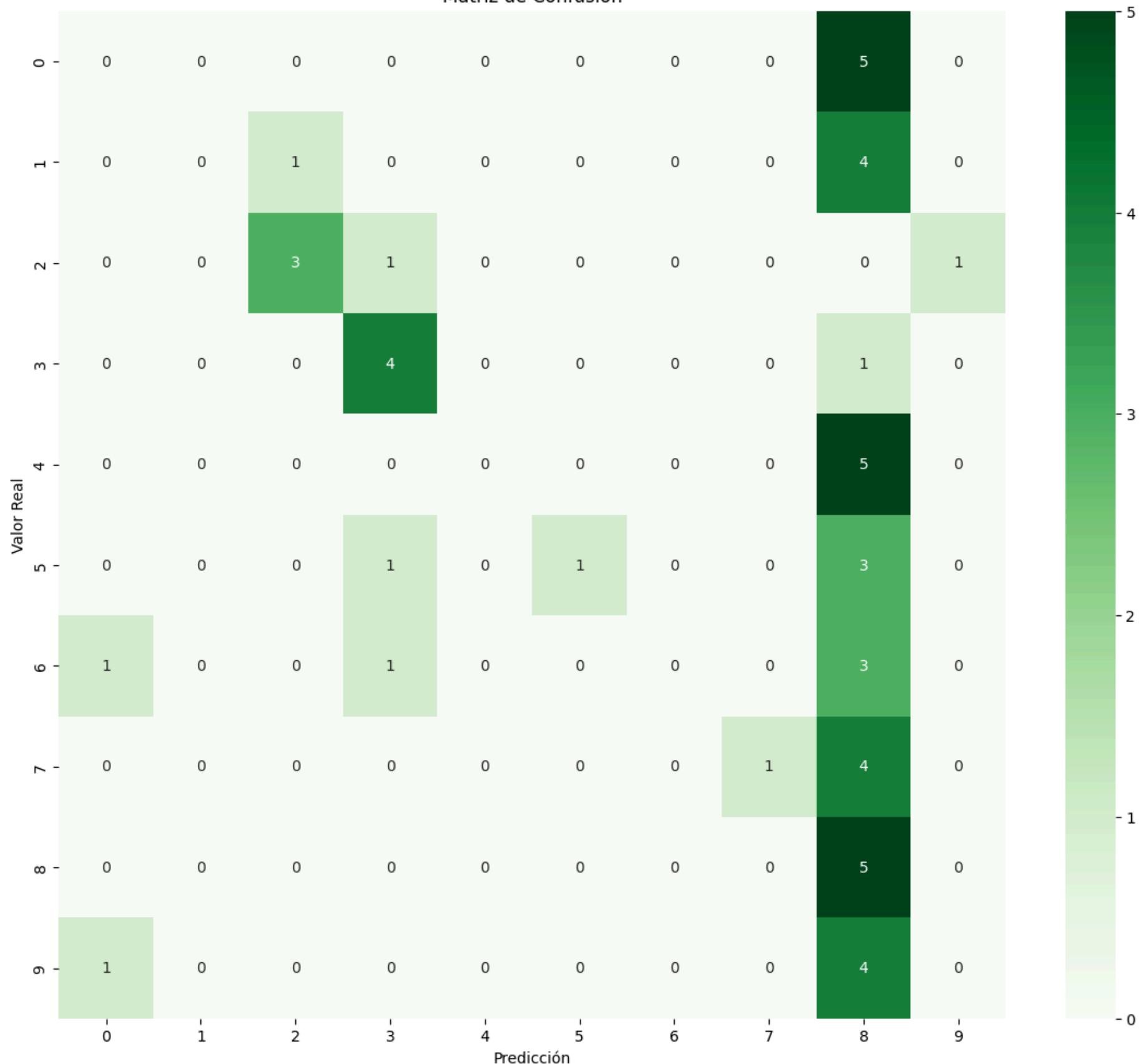


El código fue hecho en parte por Claude, pero vemos como nuestro modelo se comporta ante las imágenes que le dimos que fueron hechas a mano y se subieron al repositorio, vemos que son similares que la del base de datos, no obstante vemos que el modelo dice que la mayoría de las imágenes son un 8, cuando en realidad únicamente 5 lo son.

4. Evalúa el desempeño del modelo en dichas imágenes. Compara los resultados con los previos y comenta motivos de potencial error.

```
In [4]: accuracy=correct_count/total_count
plt.figure(figsize=(14, 12))
cm=confusion_matrix(y_true, y_pred, labels=range(10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens', xticklabels=range(10), yticklabels=range(10))
plt.xlabel('Predicción')
plt.ylabel('Valor Real')
plt.title('Matriz de Confusión')
plt.show()
print(f"Accuracy: {accuracy:.4f} ({correct_count}/{total_count})")
```

Matriz de Confusión



Accuracy: 0.2800 (14/50)

Esto también fue una parte hecha por Claude, sin embargo podemos ver que solamente le acertó a 14 de las 50 imágenes, por lo que aun no es buen modelo, o probablemente tuvimos un sobreajuste con los datos de entrenamiento.

5. Genera **3 mejoras, innovaciones, o adiciones** al sistema actual (pueden ser para cualquier etapa, desde la carga de datos, hasta los resultados de la predicción). Indica claramente qué se hizo, por qué, y si culminaron en mejoras en la predicción o en la funcionalidad del sistema.

```
In [5]: train_images=train_images.astype('float32')/255
test_images=test_images.astype('float32')/255
train_images=train_images.reshape((train_images.shape[0], 28, 28, 1))
test_images=test_images.reshape((test_images.shape[0], 28, 28, 1))
model=Sequential()
model.add(layers.Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.summary()
early_stop=EarlyStopping(monitor='val_accuracy', patience=15, restore_best_weights=True)
history=model.fit(train_images, train_labels, epochs=100, validation_split=0.4, batch_size=32, callbacks=[early_stop])
stopped_epoch=early_stop.stopped_epoch
best_epoch=stopped_epoch-early_stop.patience
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.axvline(x=best_epoch, color='r', linestyle='--', label='Early Stopping Epoch')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
```

```

plt.ylabel('Accuracy')
plt.gca().xaxis.set_major_locator(MaxNLocator(integer=True))
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.axvline(x=best_epoch, color='r', linestyle='--', label='Early Stopping Epoch')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.gca().xaxis.set_major_locator(MaxNLocator(integer=True))
plt.legend()
plt.tight_layout()
plt.show()

test_loss, test_accuracy=model.evaluate(test_images, test_labels)
print(f'Exactitud en el conjunto de prueba: {test_accuracy:.4f}')
best_val_accuracy=max(history.history['val_accuracy'])
print(f'Máxima exactitud en validación: {best_val_accuracy:.4f}')
model.save('my_model.keras')

y_pred=[]
correct_count=0
total_count=0

def convert_to_mnist_format(image_path):
    img=cv2.imread(image_path)
    gray=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray=255-gray
    resized=cv2.resize(gray, (28, 28), interpolation=cv2.INTER_AREA)
    normalized=resized.astype('float32')/255
    final_image=normalized.reshape(1, 28, 28, 1)
    return final_image

for i in range(1, 51):
    image_path=f'imagen{i}.jpg'
    final_image=convert_to_mnist_format(image_path)
    prediction=model.predict(final_image)
    predicted_label=np.argmax(prediction, axis=1)[0]
    y_pred.append(predicted_label)
    true_label=true_labels[i]
    if predicted_label==true_label:
        correct_count+=1
    total_count+=1
    print(f'Predicción: {predicted_label}, Etiqueta verdadera: {true_label}')
    plt.imshow(final_image[0].reshape(28, 28), cmap='gray')
    plt.title(f'Predicción: {predicted_label}, Etiqueta verdadera: {true_label}')
    plt.axis('off')
    plt.show()

accuracy=correct_count/total_count
plt.figure(figsize=(14, 12))
cm=confusion_matrix(y_true, y_pred, labels=range(10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens', xticklabels=range(10), yticklabels=range(10))
plt.xlabel('Predicción')
plt.ylabel('Valor Real')
plt.title('Matriz de Confusión')
plt.show()
print(f"Accuracy: {accuracy:.4f} ({correct_count}/{total_count})")

```

Model: "sequential_1"

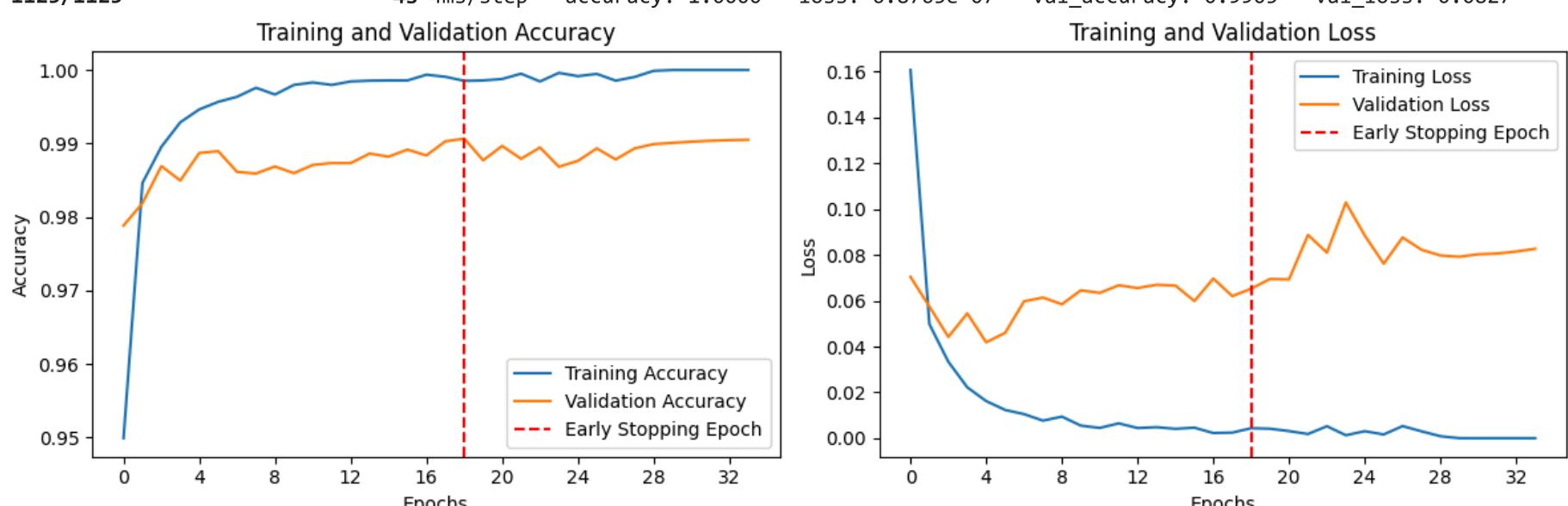
| Layer (type) | Output Shape | Param # |
|--------------------------------|--------------------|---------|
| conv2d (Conv2D) | (None, 28, 28, 32) | 320 |
| max_pooling2d (MaxPooling2D) | (None, 14, 14, 32) | 0 |
| conv2d_1 (Conv2D) | (None, 14, 14, 64) | 18,496 |
| max_pooling2d_1 (MaxPooling2D) | (None, 7, 7, 64) | 0 |
| flatten_1 (Flatten) | (None, 3136) | 0 |
| dense_4 (Dense) | (None, 128) | 401,536 |
| dense_5 (Dense) | (None, 10) | 1,290 |

Total params: 421,642 (1.61 MB)

Trainable params: 421,642 (1.61 MB)

Non-trainable params: 0 (0.00 B)

Epoch 1/100
1125/1125 4s 4ms/step - accuracy: 0.8880 - loss: 0.3561 - val_accuracy: 0.9788 - val_loss: 0.0704
Epoch 2/100
1125/1125 4s 4ms/step - accuracy: 0.9846 - loss: 0.0501 - val_accuracy: 0.9819 - val_loss: 0.0575
Epoch 3/100
1125/1125 4s 3ms/step - accuracy: 0.9897 - loss: 0.0318 - val_accuracy: 0.9869 - val_loss: 0.0442
Epoch 4/100
1125/1125 4s 3ms/step - accuracy: 0.9941 - loss: 0.0180 - val_accuracy: 0.9850 - val_loss: 0.0545
Epoch 5/100
1125/1125 4s 3ms/step - accuracy: 0.9955 - loss: 0.0142 - val_accuracy: 0.9887 - val_loss: 0.0419
Epoch 6/100
1125/1125 4s 3ms/step - accuracy: 0.9962 - loss: 0.0103 - val_accuracy: 0.9890 - val_loss: 0.0460
Epoch 7/100
1125/1125 4s 3ms/step - accuracy: 0.9966 - loss: 0.0101 - val_accuracy: 0.9861 - val_loss: 0.0598
Epoch 8/100
1125/1125 4s 3ms/step - accuracy: 0.9982 - loss: 0.0059 - val_accuracy: 0.9859 - val_loss: 0.0614
Epoch 9/100
1125/1125 4s 3ms/step - accuracy: 0.9969 - loss: 0.0087 - val_accuracy: 0.9869 - val_loss: 0.0585
Epoch 10/100
1125/1125 4s 3ms/step - accuracy: 0.9972 - loss: 0.0072 - val_accuracy: 0.9860 - val_loss: 0.0646
Epoch 11/100
1125/1125 4s 3ms/step - accuracy: 0.9986 - loss: 0.0042 - val_accuracy: 0.9871 - val_loss: 0.0634
Epoch 12/100
1125/1125 4s 3ms/step - accuracy: 0.9982 - loss: 0.0061 - val_accuracy: 0.9873 - val_loss: 0.0667
Epoch 13/100
1125/1125 4s 4ms/step - accuracy: 0.9985 - loss: 0.0041 - val_accuracy: 0.9873 - val_loss: 0.0655
Epoch 14/100
1125/1125 4s 4ms/step - accuracy: 0.9986 - loss: 0.0046 - val_accuracy: 0.9886 - val_loss: 0.0670
Epoch 15/100
1125/1125 4s 4ms/step - accuracy: 0.9986 - loss: 0.0035 - val_accuracy: 0.9882 - val_loss: 0.0666
Epoch 16/100
1125/1125 4s 4ms/step - accuracy: 0.9988 - loss: 0.0038 - val_accuracy: 0.9892 - val_loss: 0.0599
Epoch 17/100
1125/1125 4s 4ms/step - accuracy: 0.9994 - loss: 0.0018 - val_accuracy: 0.9884 - val_loss: 0.0697
Epoch 18/100
1125/1125 4s 4ms/step - accuracy: 0.9983 - loss: 0.0045 - val_accuracy: 0.9903 - val_loss: 0.0620
Epoch 19/100
1125/1125 4s 4ms/step - accuracy: 0.9982 - loss: 0.0051 - val_accuracy: 0.9907 - val_loss: 0.0653
Epoch 20/100
1125/1125 4s 4ms/step - accuracy: 0.9992 - loss: 0.0024 - val_accuracy: 0.9877 - val_loss: 0.0695
Epoch 21/100
1125/1125 4s 4ms/step - accuracy: 0.9990 - loss: 0.0024 - val_accuracy: 0.9897 - val_loss: 0.0693
Epoch 22/100
1125/1125 4s 4ms/step - accuracy: 0.9999 - loss: 5.1288e-04 - val_accuracy: 0.9879 - val_loss: 0.0887
Epoch 23/100
1125/1125 4s 4ms/step - accuracy: 0.9984 - loss: 0.0059 - val_accuracy: 0.9895 - val_loss: 0.0810
Epoch 24/100
1125/1125 4s 4ms/step - accuracy: 0.9995 - loss: 0.0017 - val_accuracy: 0.9868 - val_loss: 0.1029
Epoch 25/100
1125/1125 4s 4ms/step - accuracy: 0.9990 - loss: 0.0043 - val_accuracy: 0.9876 - val_loss: 0.0885
Epoch 26/100
1125/1125 4s 4ms/step - accuracy: 0.9995 - loss: 0.0017 - val_accuracy: 0.9893 - val_loss: 0.0762
Epoch 27/100
1125/1125 4s 3ms/step - accuracy: 0.9990 - loss: 0.0037 - val_accuracy: 0.9878 - val_loss: 0.0876
Epoch 28/100
1125/1125 4s 3ms/step - accuracy: 0.9990 - loss: 0.0031 - val_accuracy: 0.9893 - val_loss: 0.0822
Epoch 29/100
1125/1125 4s 4ms/step - accuracy: 0.9998 - loss: 0.0016 - val_accuracy: 0.9899 - val_loss: 0.0797
Epoch 30/100
1125/1125 4s 4ms/step - accuracy: 1.0000 - loss: 1.2711e-05 - val_accuracy: 0.9901 - val_loss: 0.0792
Epoch 31/100
1125/1125 4s 4ms/step - accuracy: 1.0000 - loss: 6.3595e-06 - val_accuracy: 0.9902 - val_loss: 0.0803
Epoch 32/100
1125/1125 4s 4ms/step - accuracy: 1.0000 - loss: 2.2963e-06 - val_accuracy: 0.9904 - val_loss: 0.0806
Epoch 33/100
1125/1125 4s 4ms/step - accuracy: 1.0000 - loss: 1.0863e-06 - val_accuracy: 0.9905 - val_loss: 0.0815
Epoch 34/100
1125/1125 4s 4ms/step - accuracy: 1.0000 - loss: 6.8765e-07 - val_accuracy: 0.9905 - val_loss: 0.0827



313/313 ————— 0s 1ms/step - accuracy: 0.9875 - loss: 0.0571

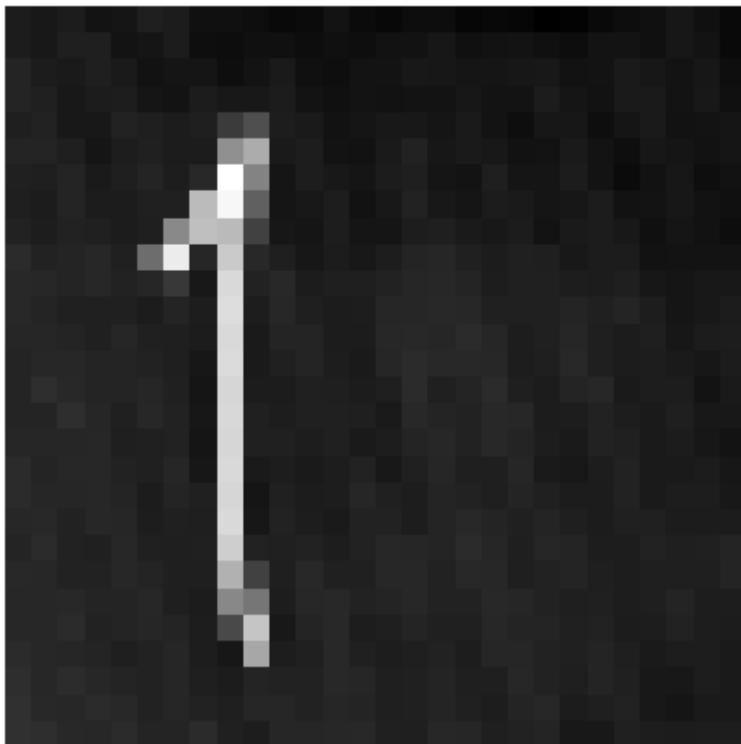
Exactitud en el conjunto de prueba: 0.9904

Máxima exactitud en validación: 0.9907

1/1 ————— 0s 42ms/step

Predicción: 6, Etiqueta verdadera: 1

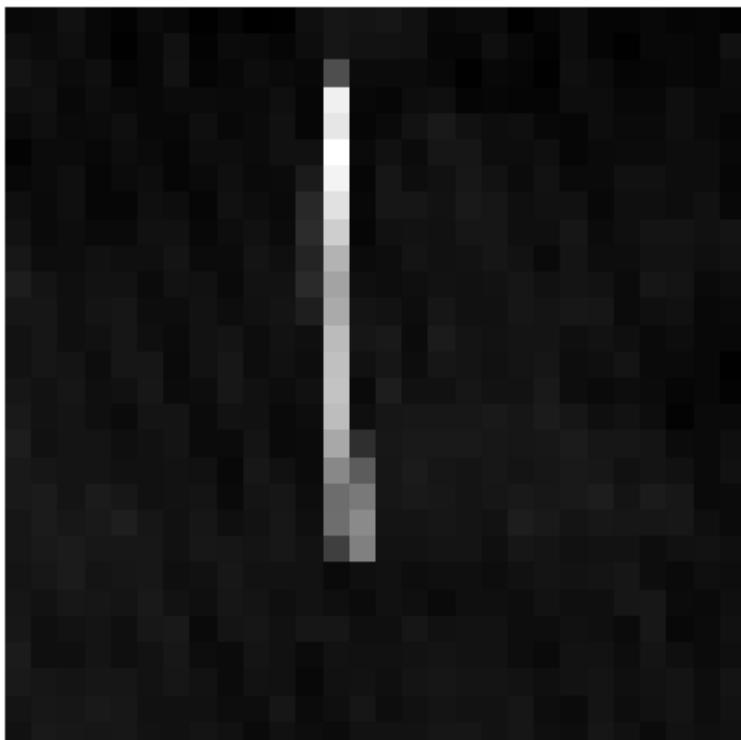
Predicción: 6, Etiqueta verdadera: 1



1/1 ————— 0s 21ms/step

Predicción: 1, Etiqueta verdadera: 1

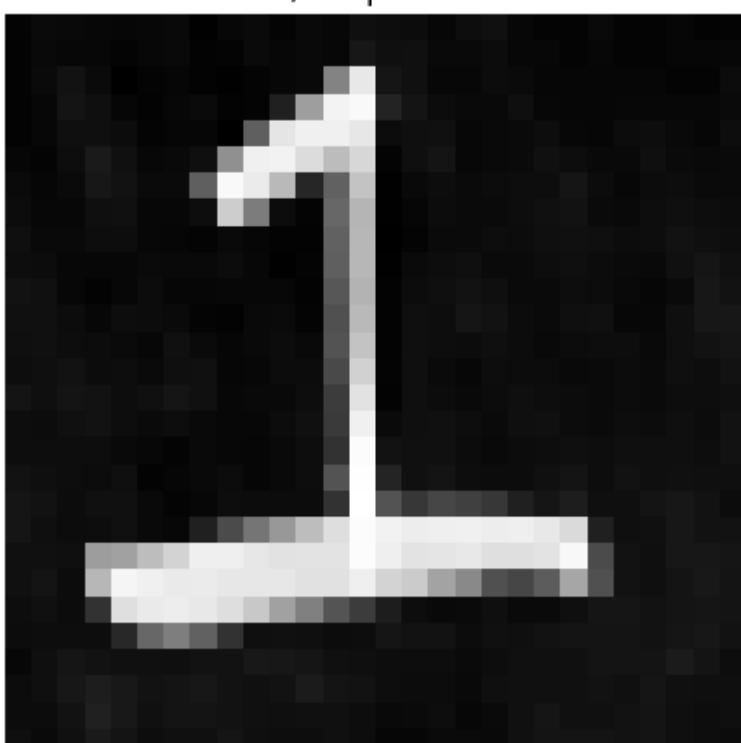
Predicción: 1, Etiqueta verdadera: 1



1/1 ————— 0s 20ms/step

Predicción: 1, Etiqueta verdadera: 1

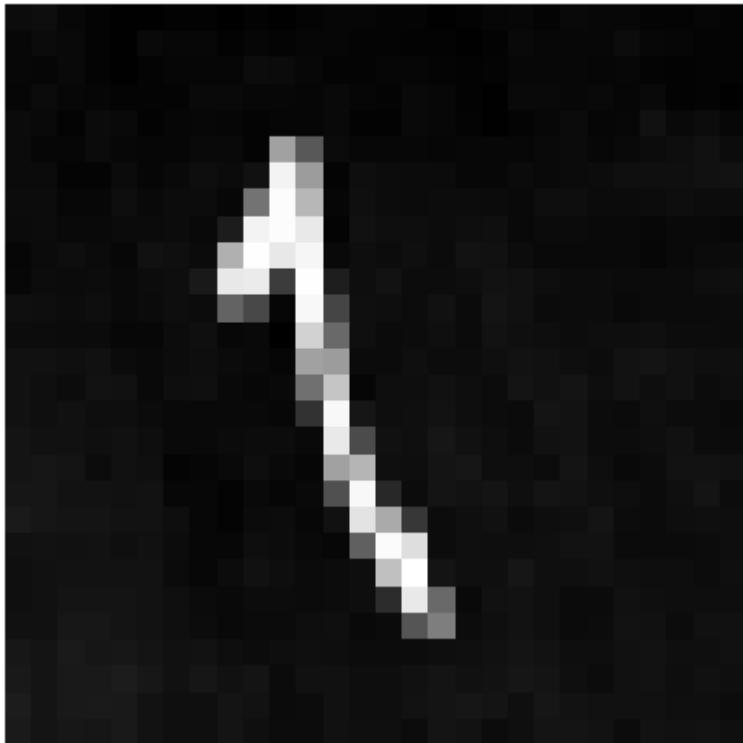
Predicción: 1, Etiqueta verdadera: 1



1/1 ————— 0s 20ms/step

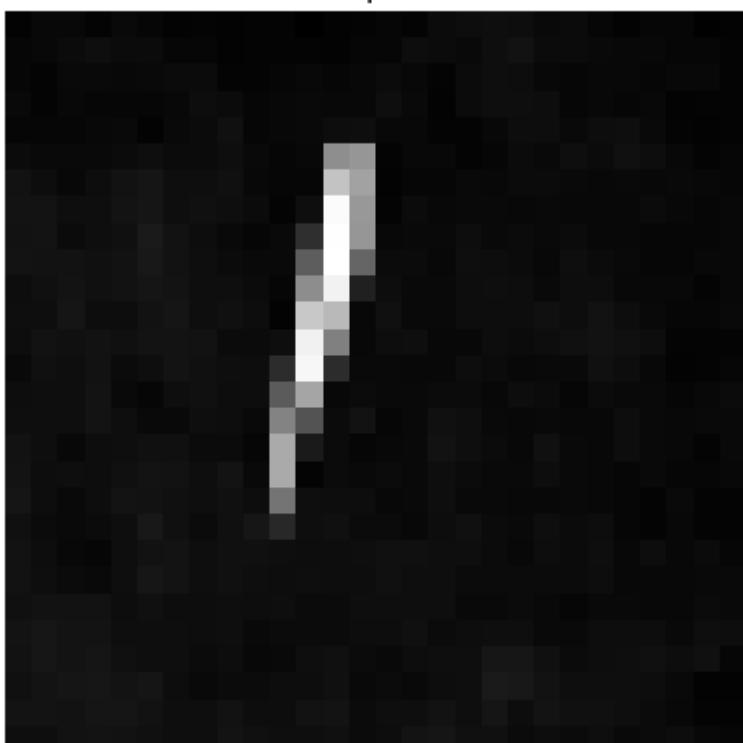
Predicción: 1, Etiqueta verdadera: 1

Predicción: 1, Etiqueta verdadera: 1



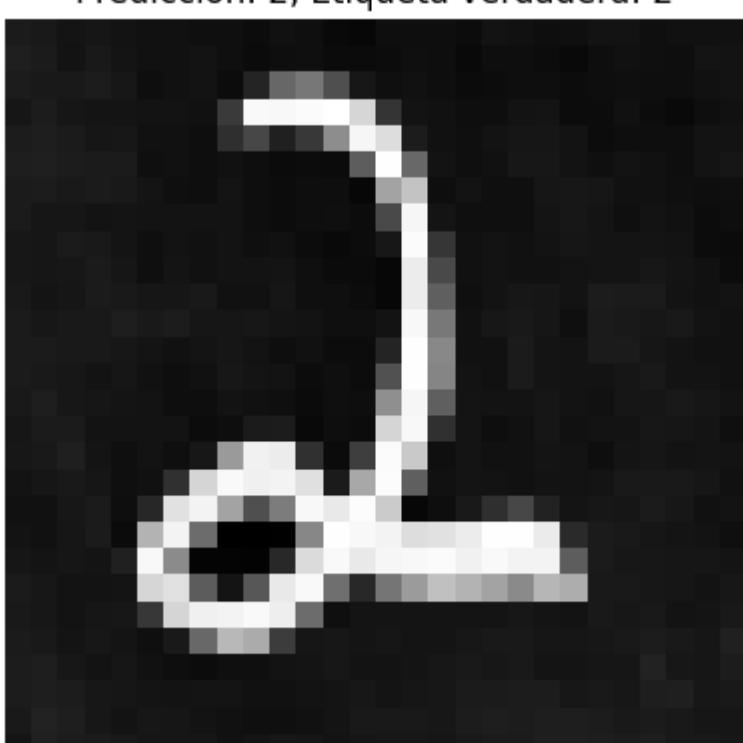
1/1 ————— 0s 20ms/step
Predicción: 4, Etiqueta verdadera: 1

Predicción: 4, Etiqueta verdadera: 1



1/1 ————— 0s 20ms/step
Predicción: 2, Etiqueta verdadera: 2

Predicción: 2, Etiqueta verdadera: 2



1/1 ————— 0s 20ms/step
Predicción: 2, Etiqueta verdadera: 2

Predicción: 2, Etiqueta verdadera: 2



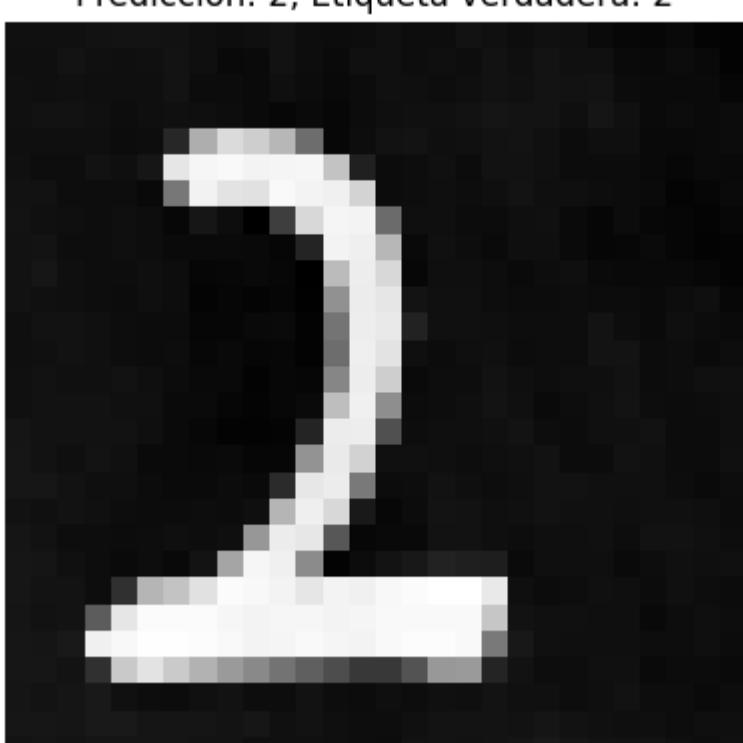
1/1 ————— 0s 28ms/step
Predicción: 2, Etiqueta verdadera: 2

Predicción: 2, Etiqueta verdadera: 2



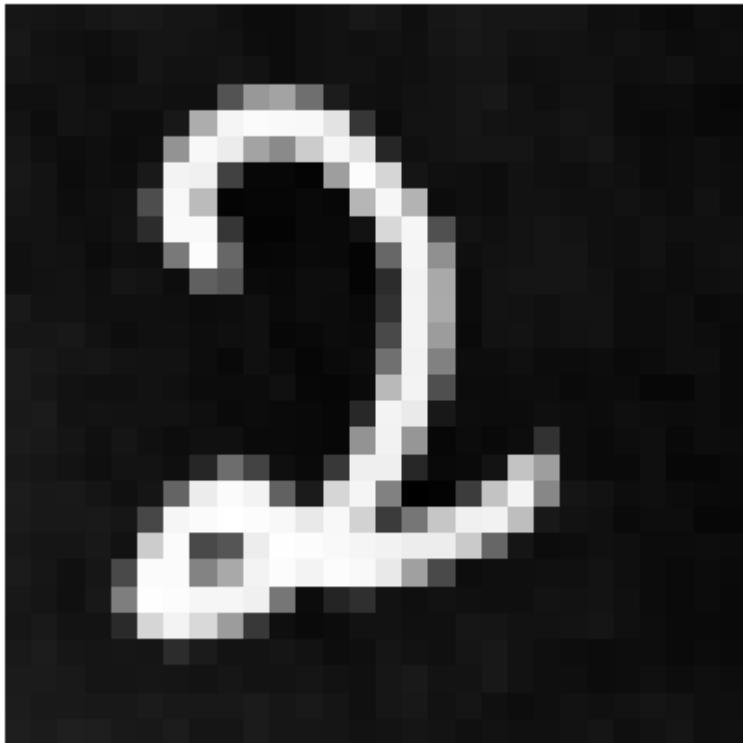
1/1 ————— 0s 21ms/step
Predicción: 2, Etiqueta verdadera: 2

Predicción: 2, Etiqueta verdadera: 2



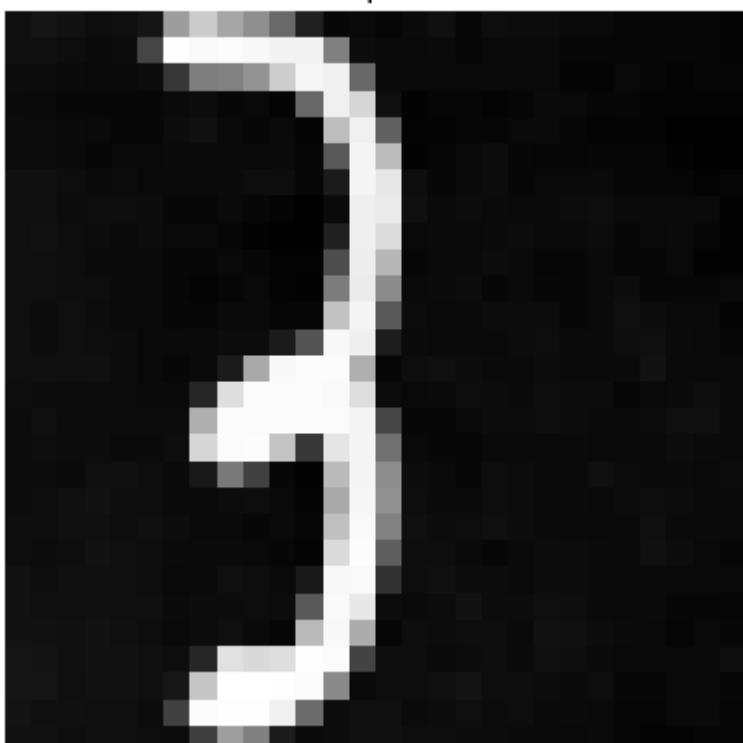
1/1 ————— 0s 20ms/step
Predicción: 2, Etiqueta verdadera: 2

Predicción: 2, Etiqueta verdadera: 2



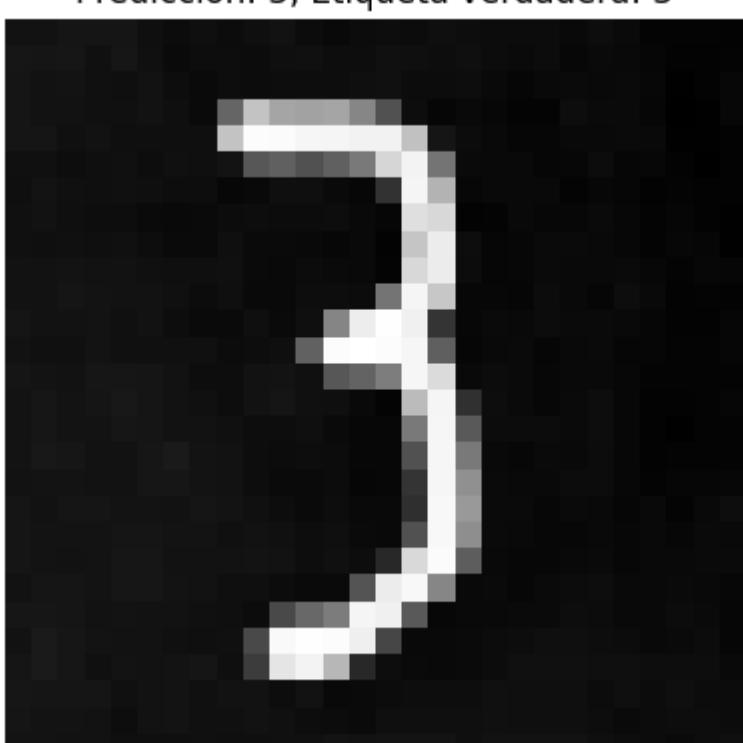
1/1 ————— 0s 26ms/step
Predicción: 1, Etiqueta verdadera: 3

Predicción: 1, Etiqueta verdadera: 3



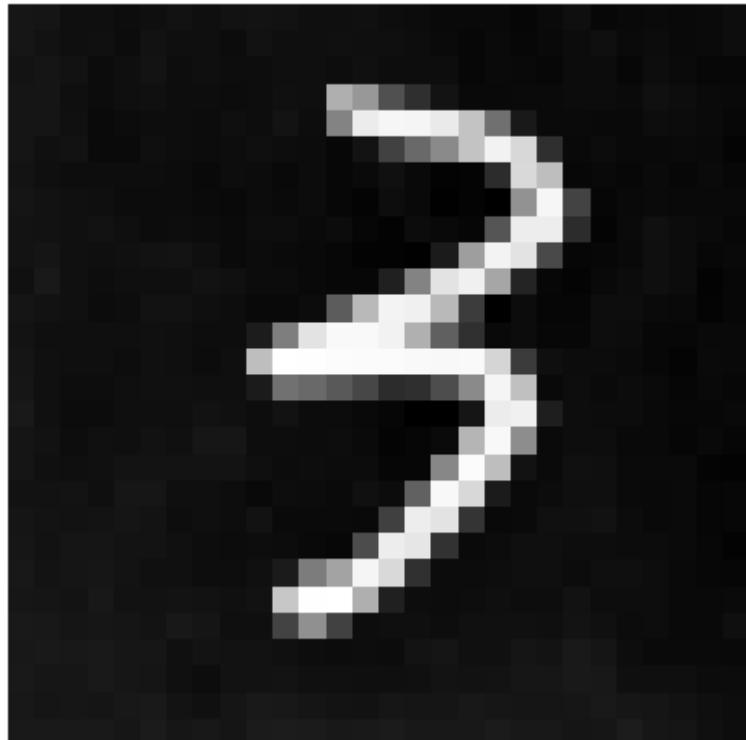
1/1 ————— 0s 20ms/step
Predicción: 3, Etiqueta verdadera: 3

Predicción: 3, Etiqueta verdadera: 3



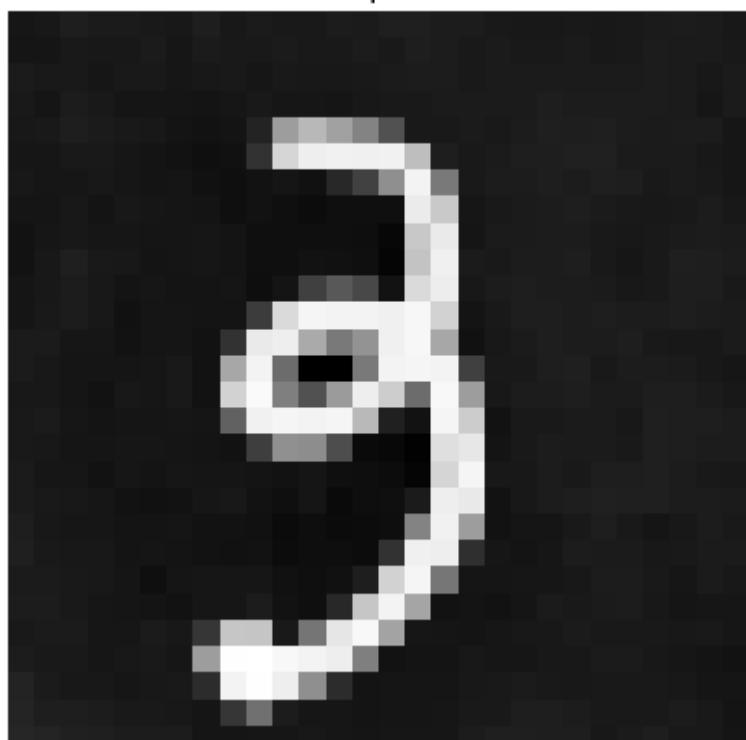
1/1 ————— 0s 20ms/step
Predicción: 3, Etiqueta verdadera: 3

Predicción: 3, Etiqueta verdadera: 3



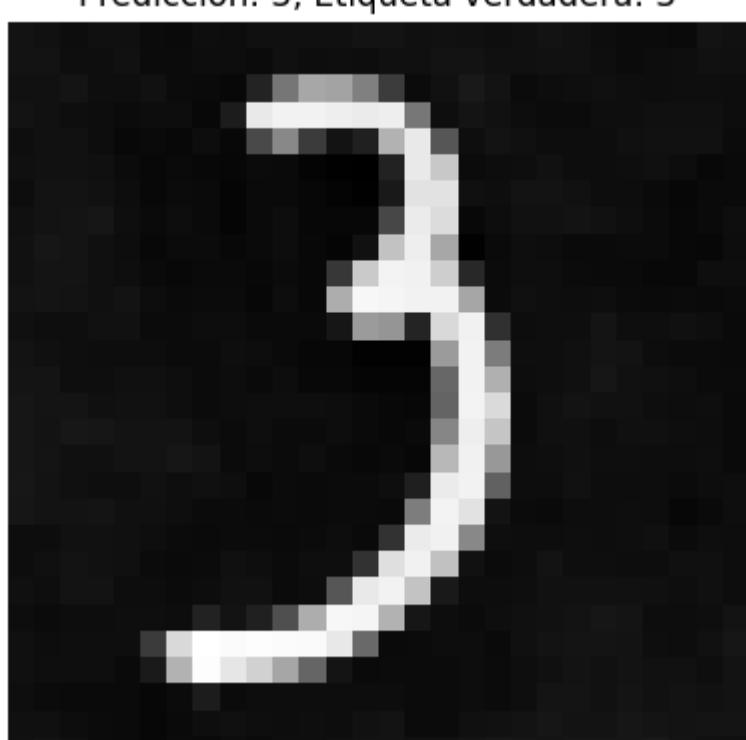
1/1 ————— 0s 19ms/step
Predicción: 3, Etiqueta verdadera: 3

Predicción: 3, Etiqueta verdadera: 3



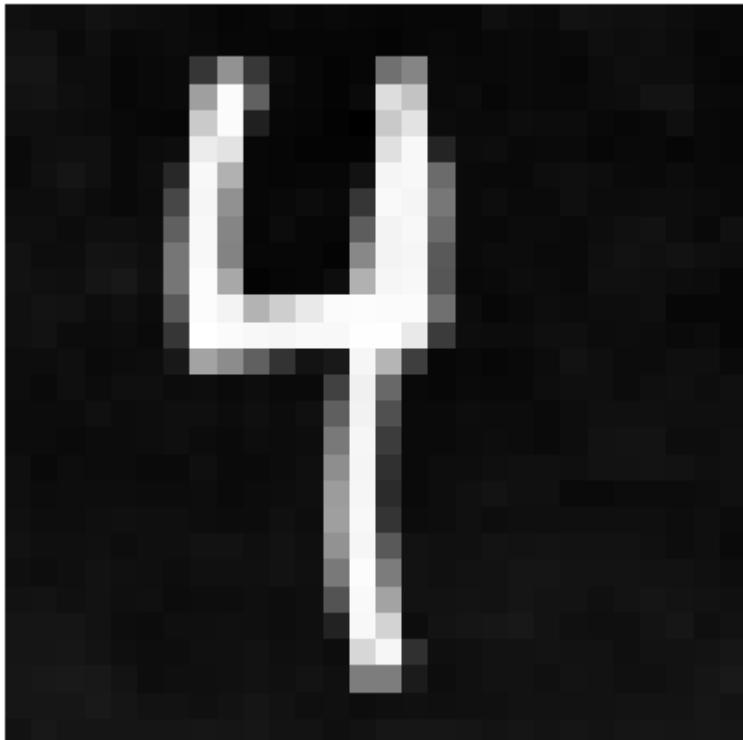
1/1 ————— 0s 21ms/step
Predicción: 3, Etiqueta verdadera: 3

Predicción: 3, Etiqueta verdadera: 3



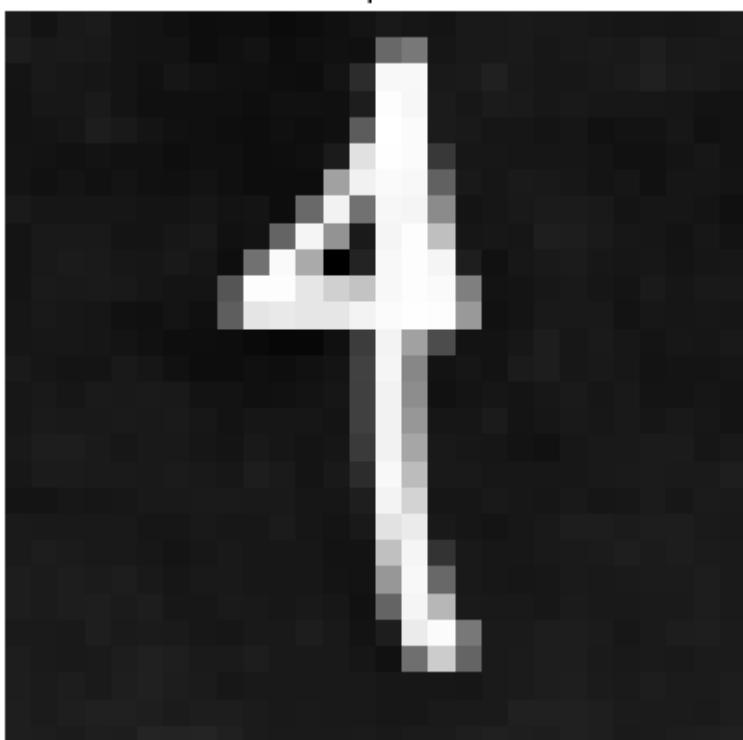
1/1 ————— 0s 22ms/step
Predicción: 1, Etiqueta verdadera: 4

Predicción: 1, Etiqueta verdadera: 4



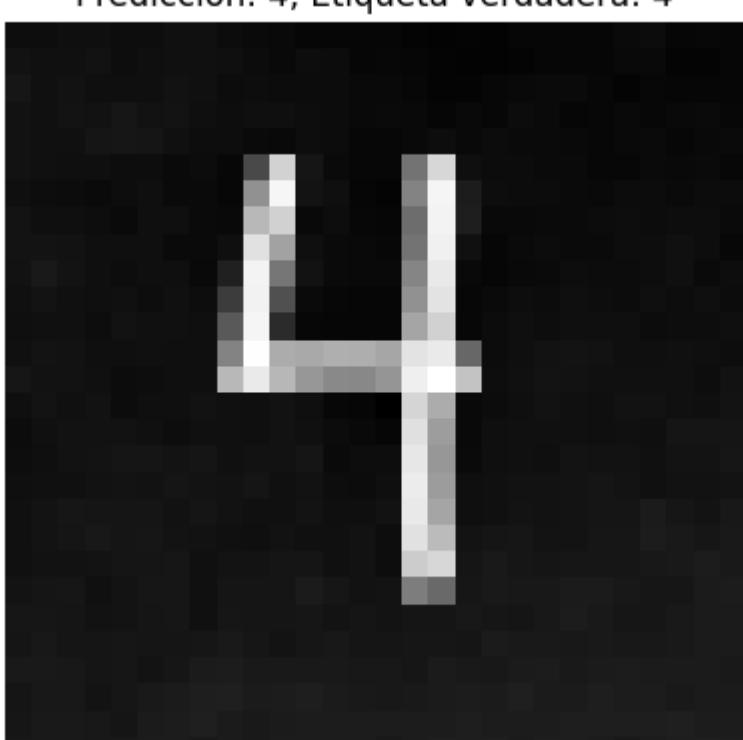
1/1 ————— 0s 21ms/step
Predicción: 1, Etiqueta verdadera: 4

Predicción: 1, Etiqueta verdadera: 4



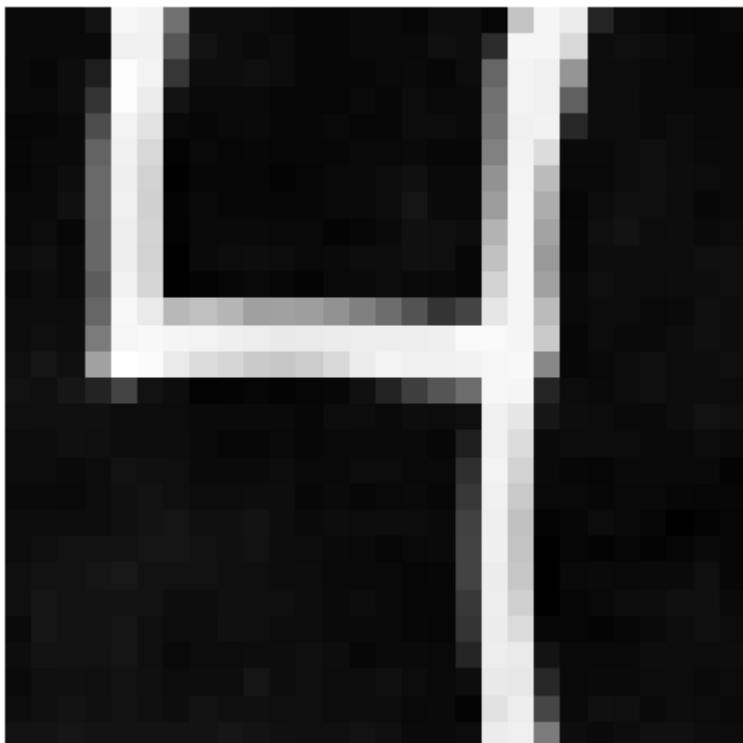
1/1 ————— 0s 22ms/step
Predicción: 4, Etiqueta verdadera: 4

Predicción: 4, Etiqueta verdadera: 4



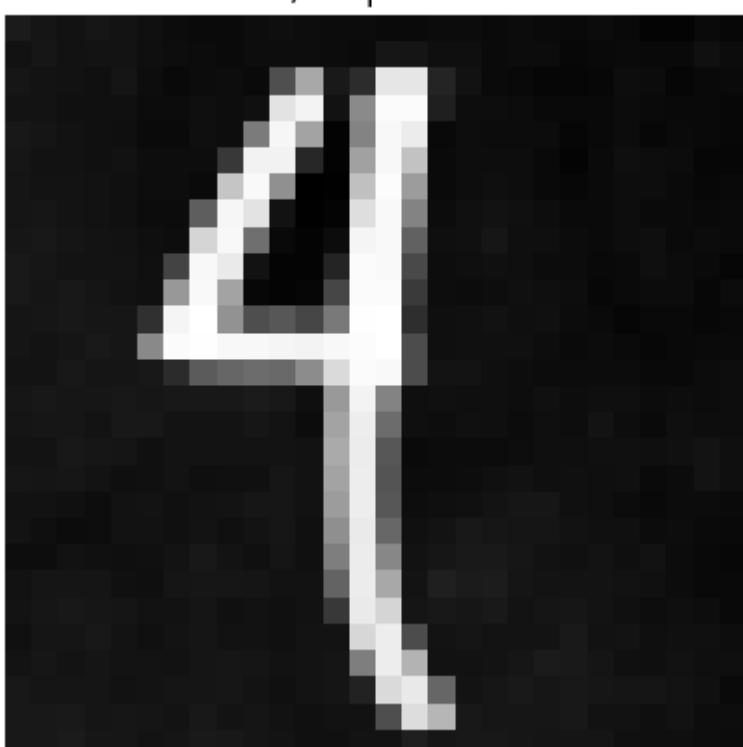
1/1 ————— 0s 20ms/step
Predicción: 4, Etiqueta verdadera: 4

Predicción: 4, Etiqueta verdadera: 4



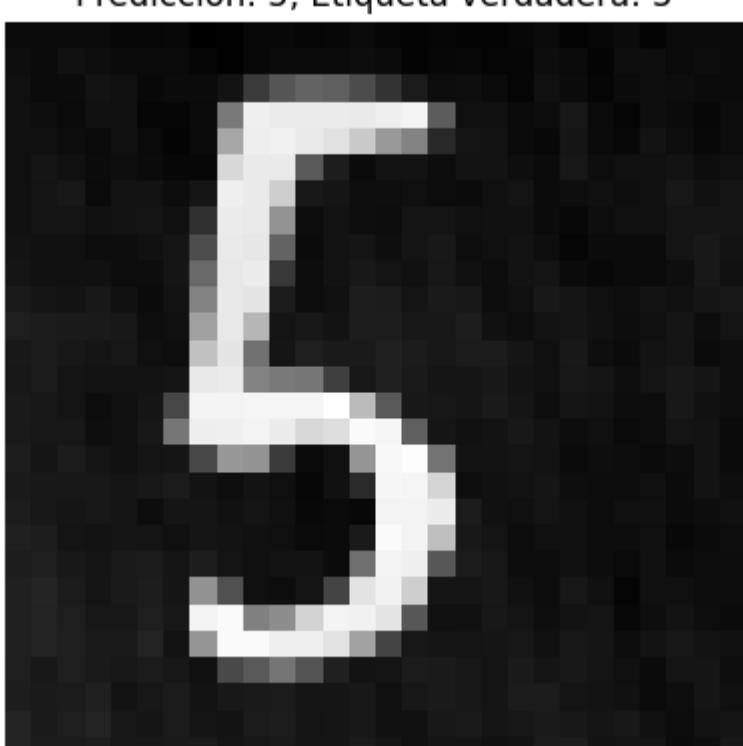
1/1 ————— 0s 20ms/step
Predicción: 1, Etiqueta verdadera: 4

Predicción: 1, Etiqueta verdadera: 4



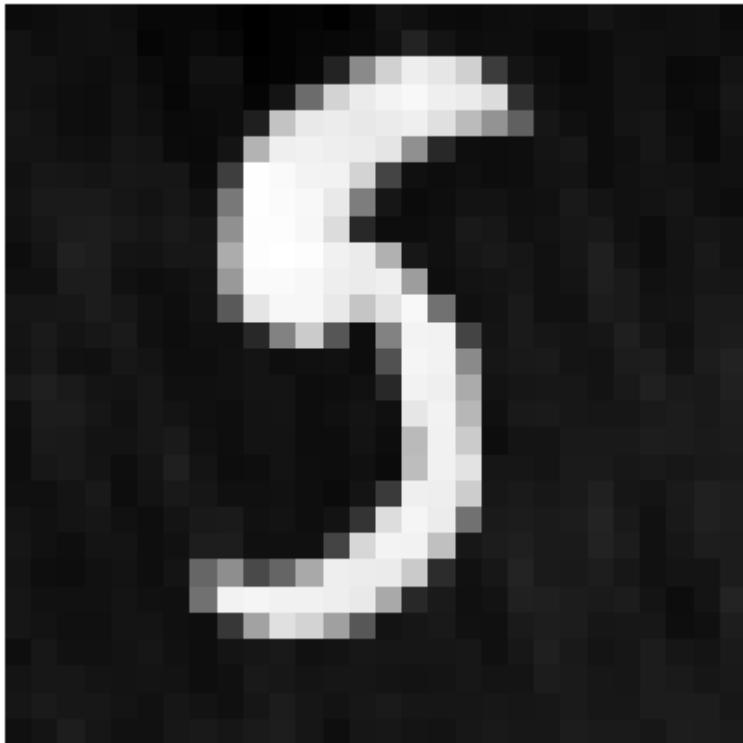
1/1 ————— 0s 27ms/step
Predicción: 5, Etiqueta verdadera: 5

Predicción: 5, Etiqueta verdadera: 5



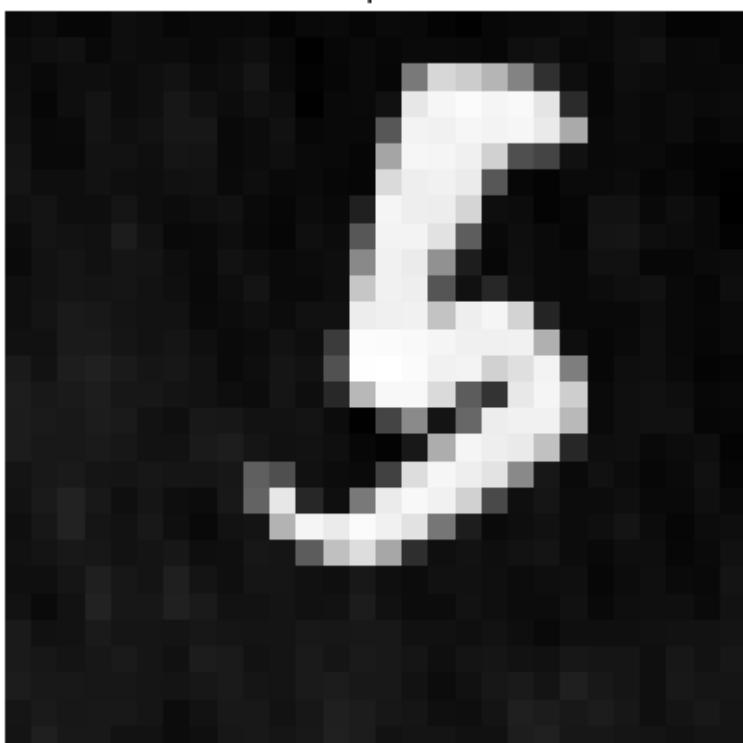
1/1 ————— 0s 19ms/step
Predicción: 5, Etiqueta verdadera: 5

Predicción: 5, Etiqueta verdadera: 5



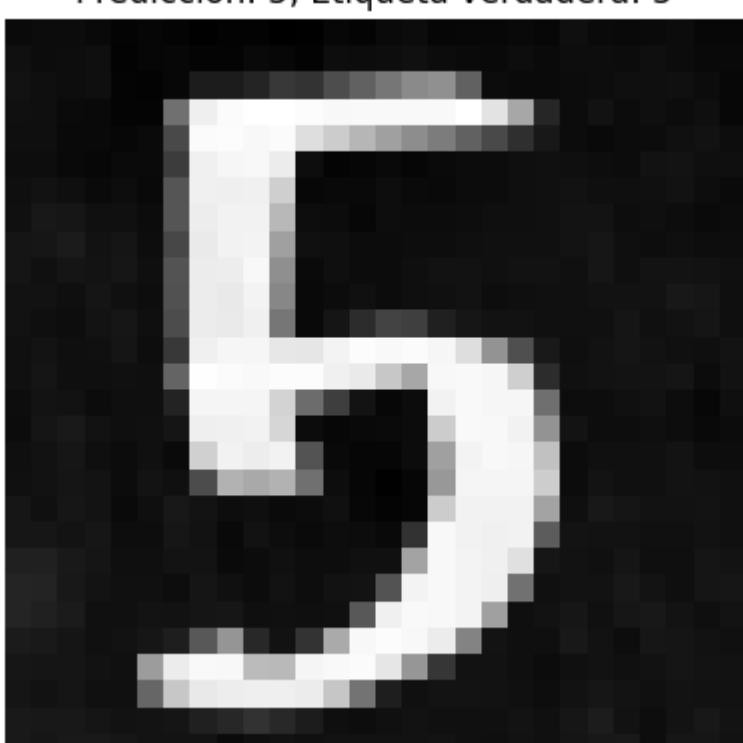
1/1 ————— 0s 20ms/step
Predicción: 6, Etiqueta verdadera: 5

Predicción: 6, Etiqueta verdadera: 5



1/1 ————— 0s 20ms/step
Predicción: 5, Etiqueta verdadera: 5

Predicción: 5, Etiqueta verdadera: 5



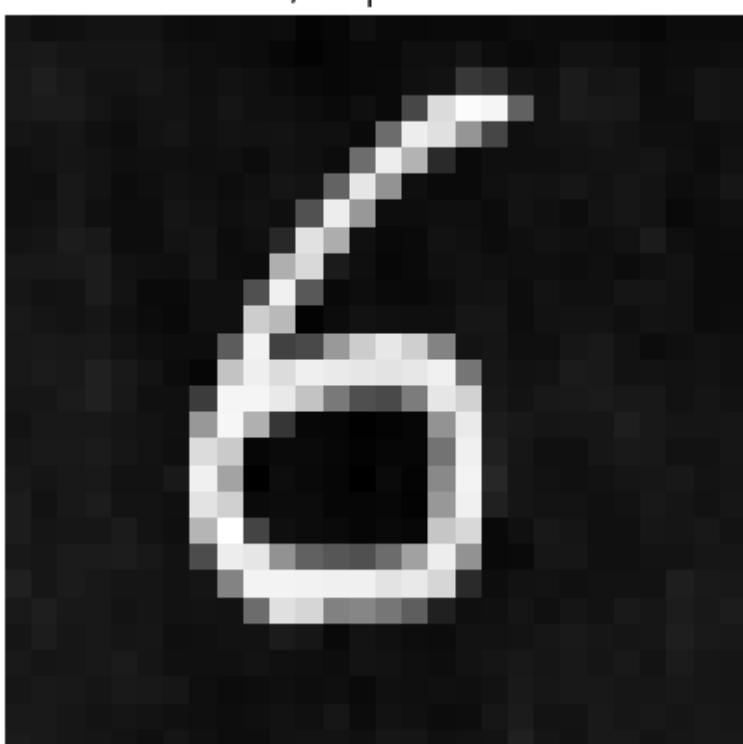
1/1 ————— 0s 30ms/step
Predicción: 5, Etiqueta verdadera: 5

Predicción: 5, Etiqueta verdadera: 5



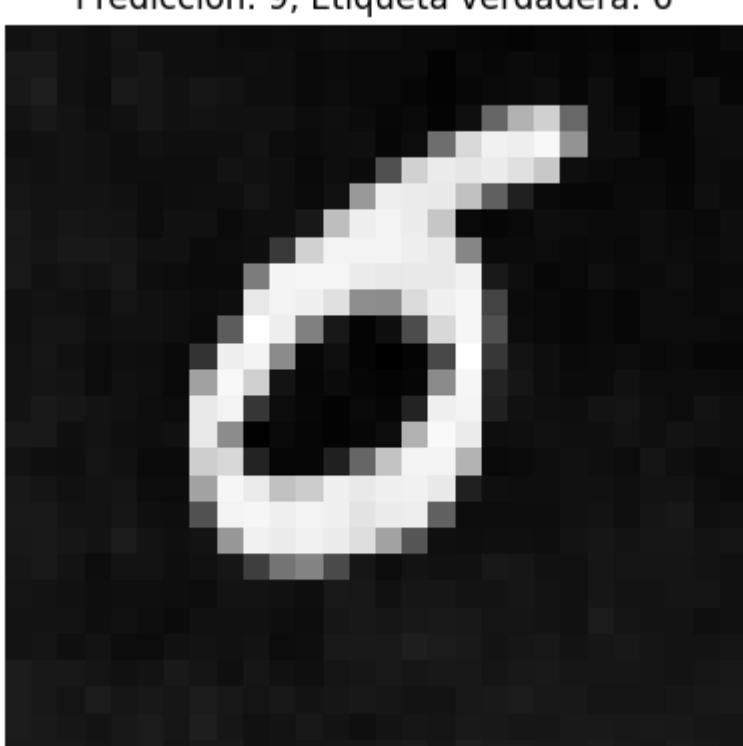
1/1 ————— 0s 19ms/step
Predicción: 6, Etiqueta verdadera: 6

Predicción: 6, Etiqueta verdadera: 6



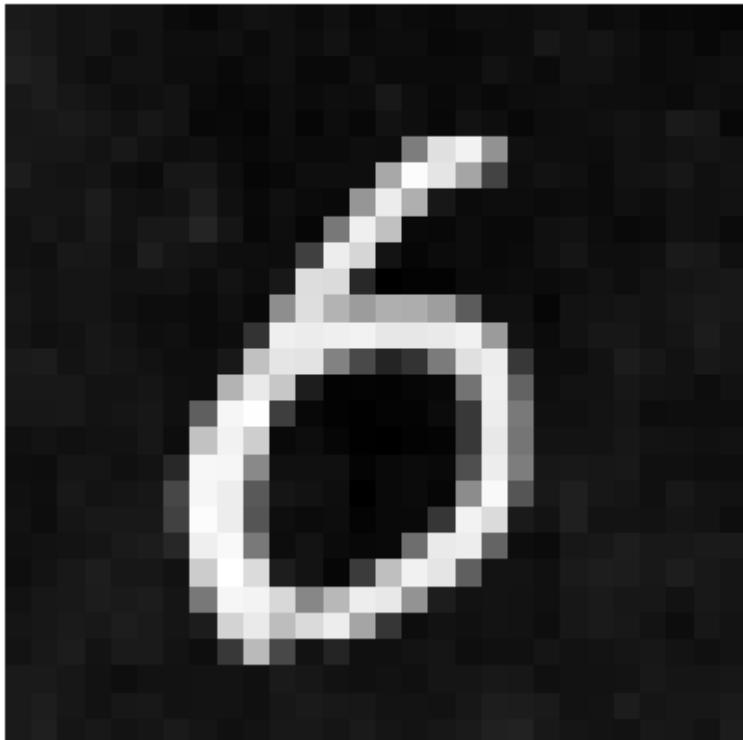
1/1 ————— 0s 20ms/step
Predicción: 9, Etiqueta verdadera: 6

Predicción: 9, Etiqueta verdadera: 6



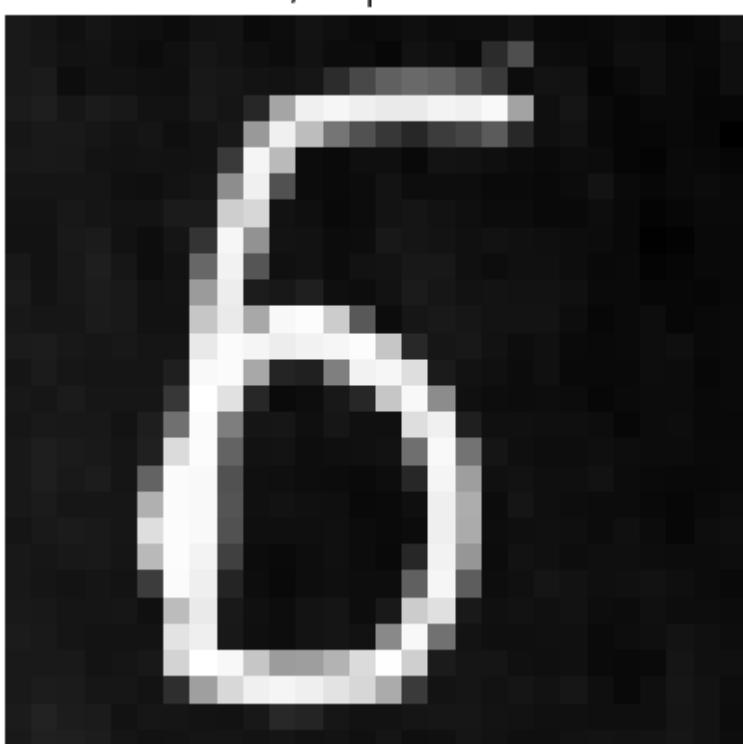
1/1 ————— 0s 26ms/step
Predicción: 6, Etiqueta verdadera: 6

Predicción: 6, Etiqueta verdadera: 6



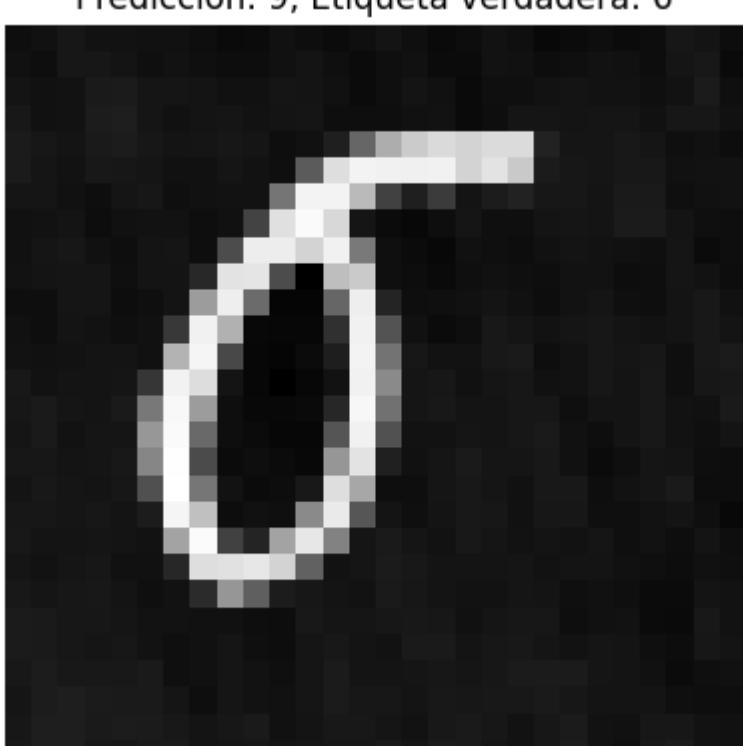
1/1 ————— 0s 19ms/step
Predicción: 5, Etiqueta verdadera: 6

Predicción: 5, Etiqueta verdadera: 6



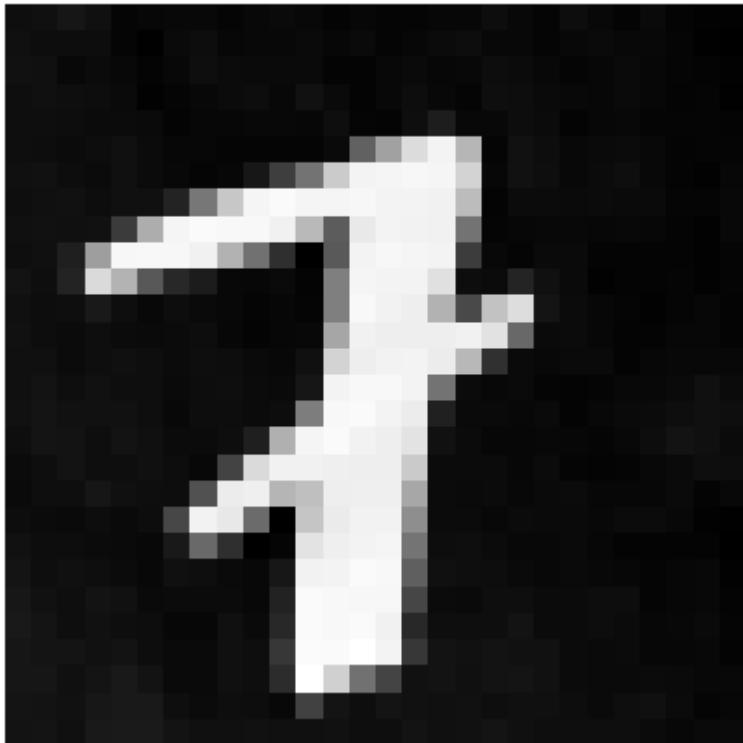
1/1 ————— 0s 18ms/step
Predicción: 9, Etiqueta verdadera: 6

Predicción: 9, Etiqueta verdadera: 6



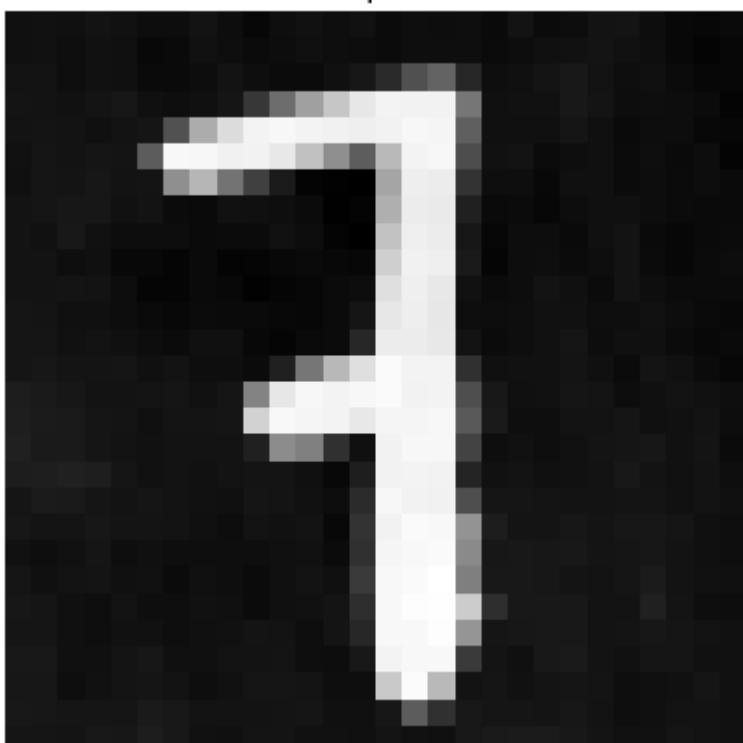
1/1 ————— 0s 19ms/step
Predicción: 7, Etiqueta verdadera: 7

Predicción: 7, Etiqueta verdadera: 7



1/1 ————— 0s 20ms/step
Predicción: 3, Etiqueta verdadera: 7

Predicción: 3, Etiqueta verdadera: 7



1/1 ————— 0s 20ms/step
Predicción: 7, Etiqueta verdadera: 7

Predicción: 7, Etiqueta verdadera: 7



1/1 ————— 0s 20ms/step
Predicción: 7, Etiqueta verdadera: 7

Predicción: 7, Etiqueta verdadera: 7



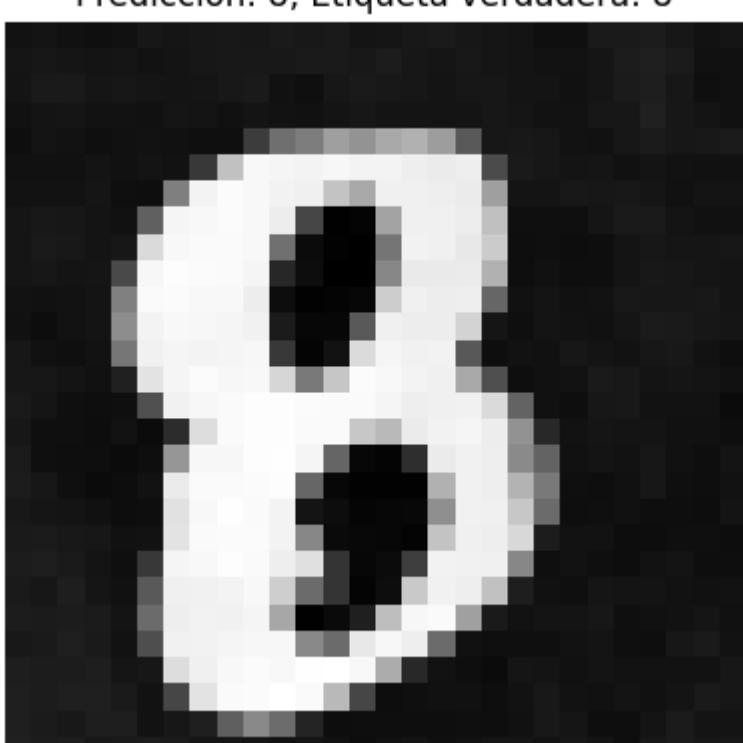
1/1 ————— 0s 20ms/step
Predicción: 2, Etiqueta verdadera: 7

Predicción: 2, Etiqueta verdadera: 7



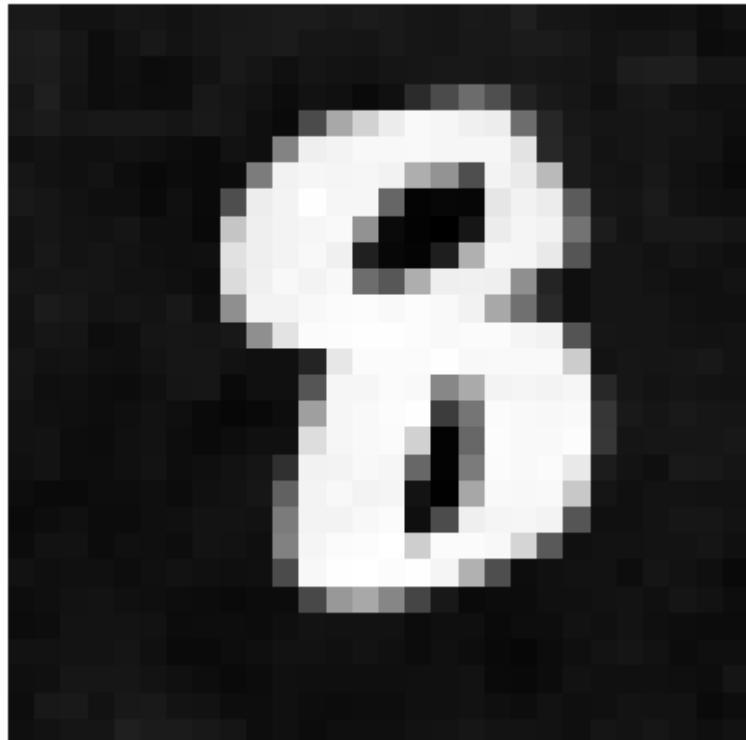
1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 8

Predicción: 8, Etiqueta verdadera: 8



1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 8

Predicción: 8, Etiqueta verdadera: 8



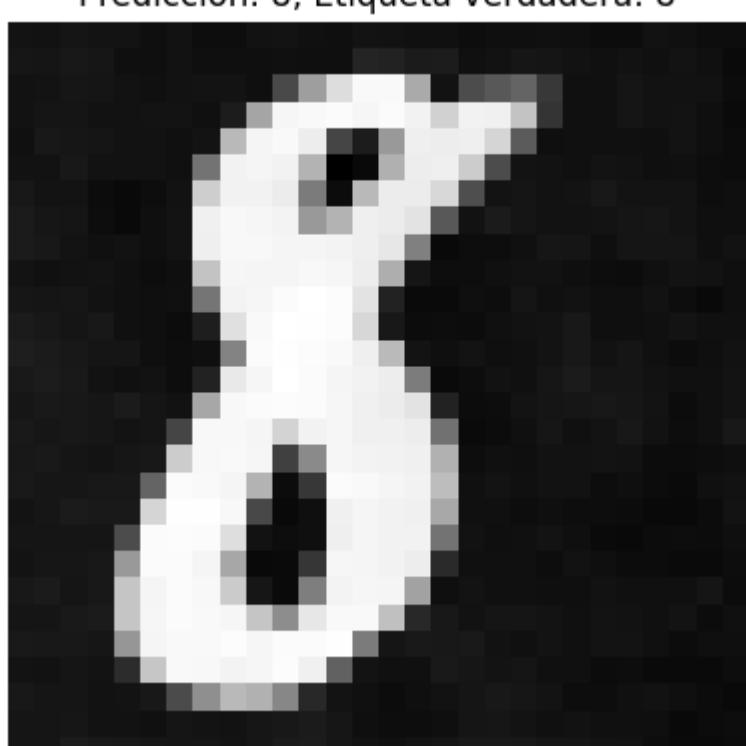
1/1 ————— 0s 19ms/step
Predicción: 8, Etiqueta verdadera: 8

Predicción: 8, Etiqueta verdadera: 8



1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 8

Predicción: 8, Etiqueta verdadera: 8



1/1 ————— 0s 21ms/step
Predicción: 8, Etiqueta verdadera: 8

Predicción: 8, Etiqueta verdadera: 8



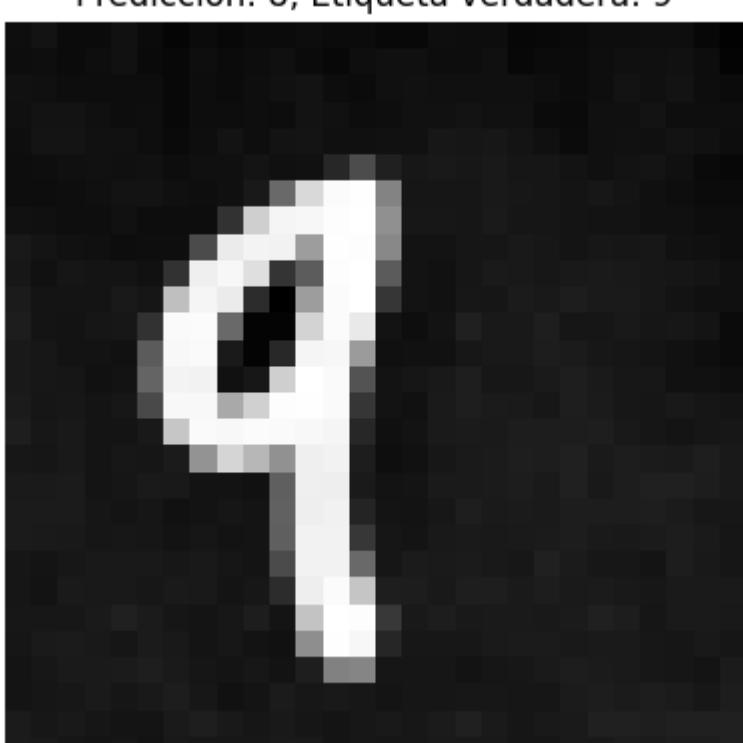
1/1 ————— 0s 20ms/step
Predicción: 0, Etiqueta verdadera: 9

Predicción: 0, Etiqueta verdadera: 9



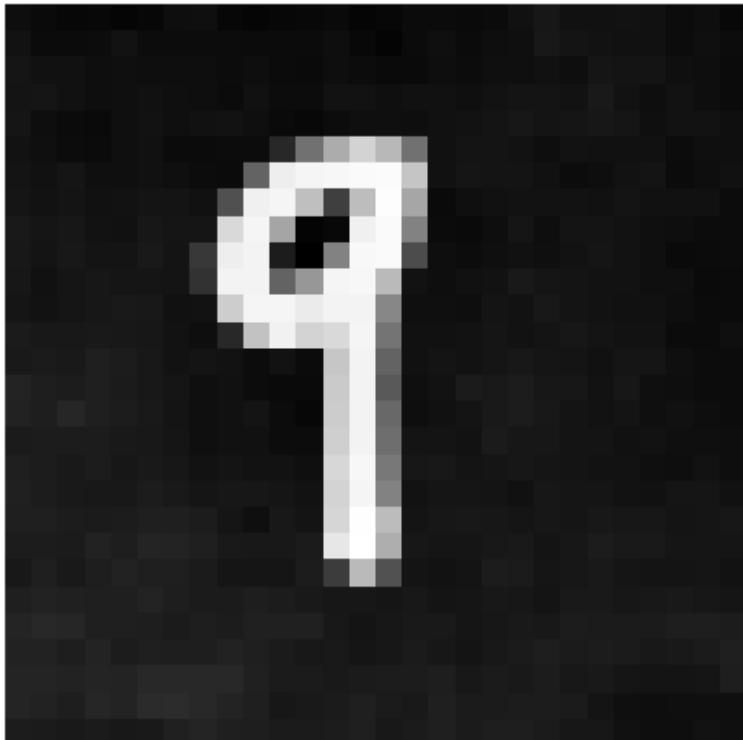
1/1 ————— 0s 20ms/step
Predicción: 8, Etiqueta verdadera: 9

Predicción: 8, Etiqueta verdadera: 9



1/1 ————— 0s 21ms/step
Predicción: 9, Etiqueta verdadera: 9

Predicción: 9, Etiqueta verdadera: 9



1/1 ━━━━━━ 0s 21ms/step
Predicción: 8, Etiqueta verdadera: 9

Predicción: 8, Etiqueta verdadera: 9



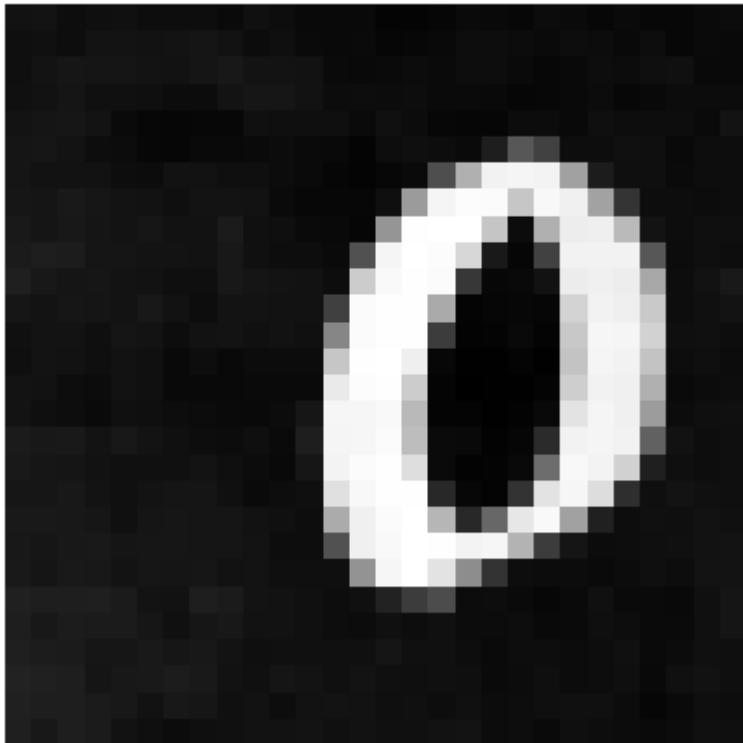
1/1 ━━━━━━ 0s 21ms/step
Predicción: 2, Etiqueta verdadera: 9

Predicción: 2, Etiqueta verdadera: 9



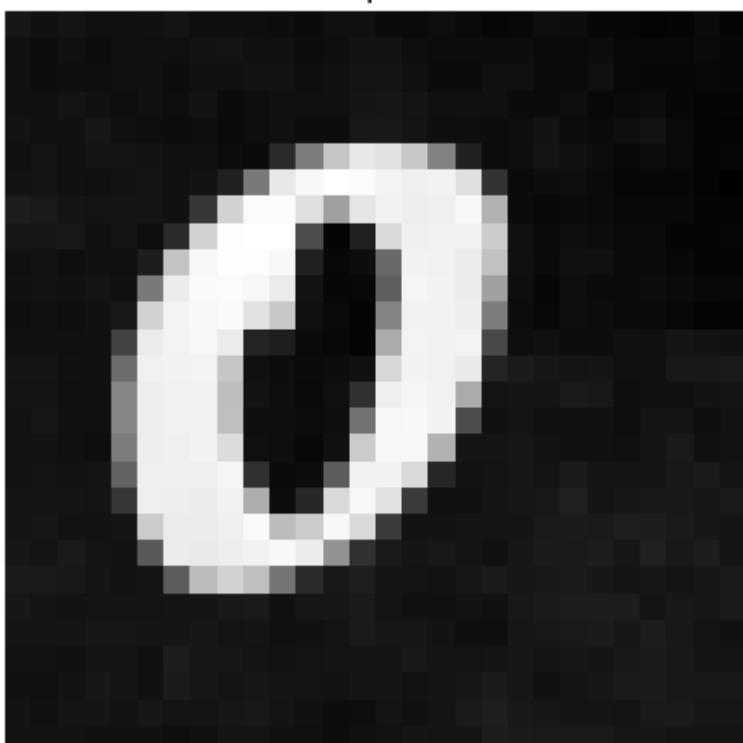
1/1 ━━━━━━ 0s 21ms/step
Predicción: 0, Etiqueta verdadera: 0

Predicción: 0, Etiqueta verdadera: 0



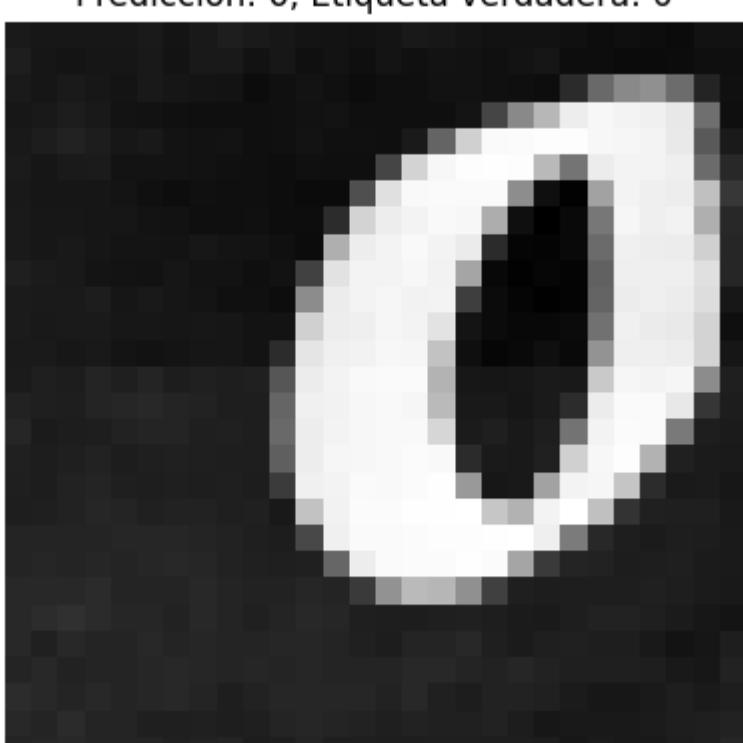
1/1 ————— 0s 20ms/step
Predicción: 0, Etiqueta verdadera: 0

Predicción: 0, Etiqueta verdadera: 0



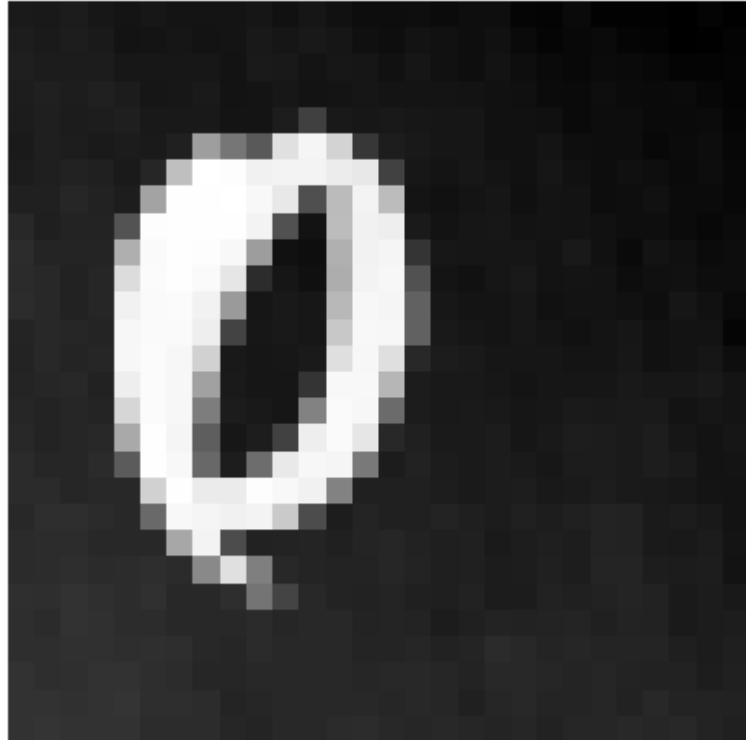
1/1 ————— 0s 20ms/step
Predicción: 6, Etiqueta verdadera: 0

Predicción: 6, Etiqueta verdadera: 0



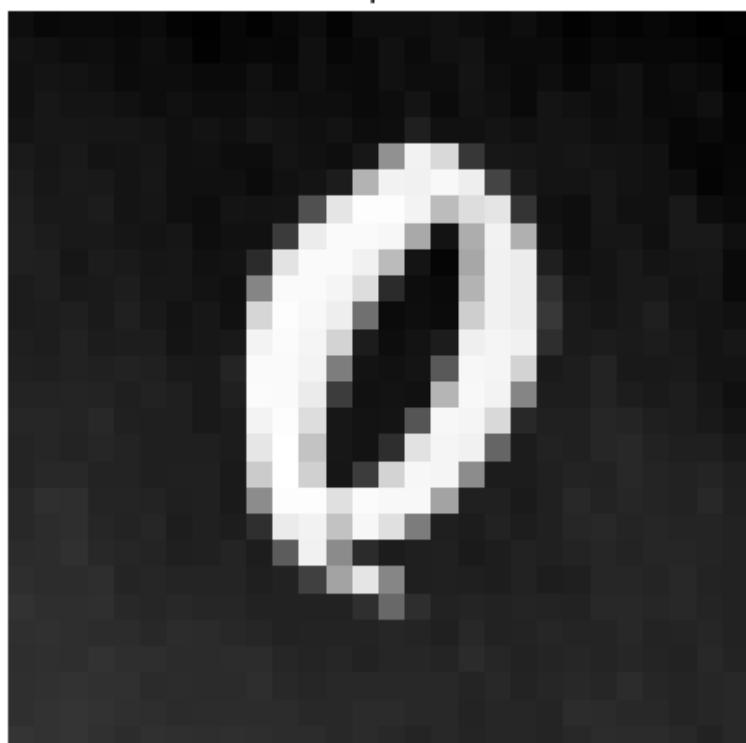
1/1 ————— 0s 20ms/step
Predicción: 0, Etiqueta verdadera: 0

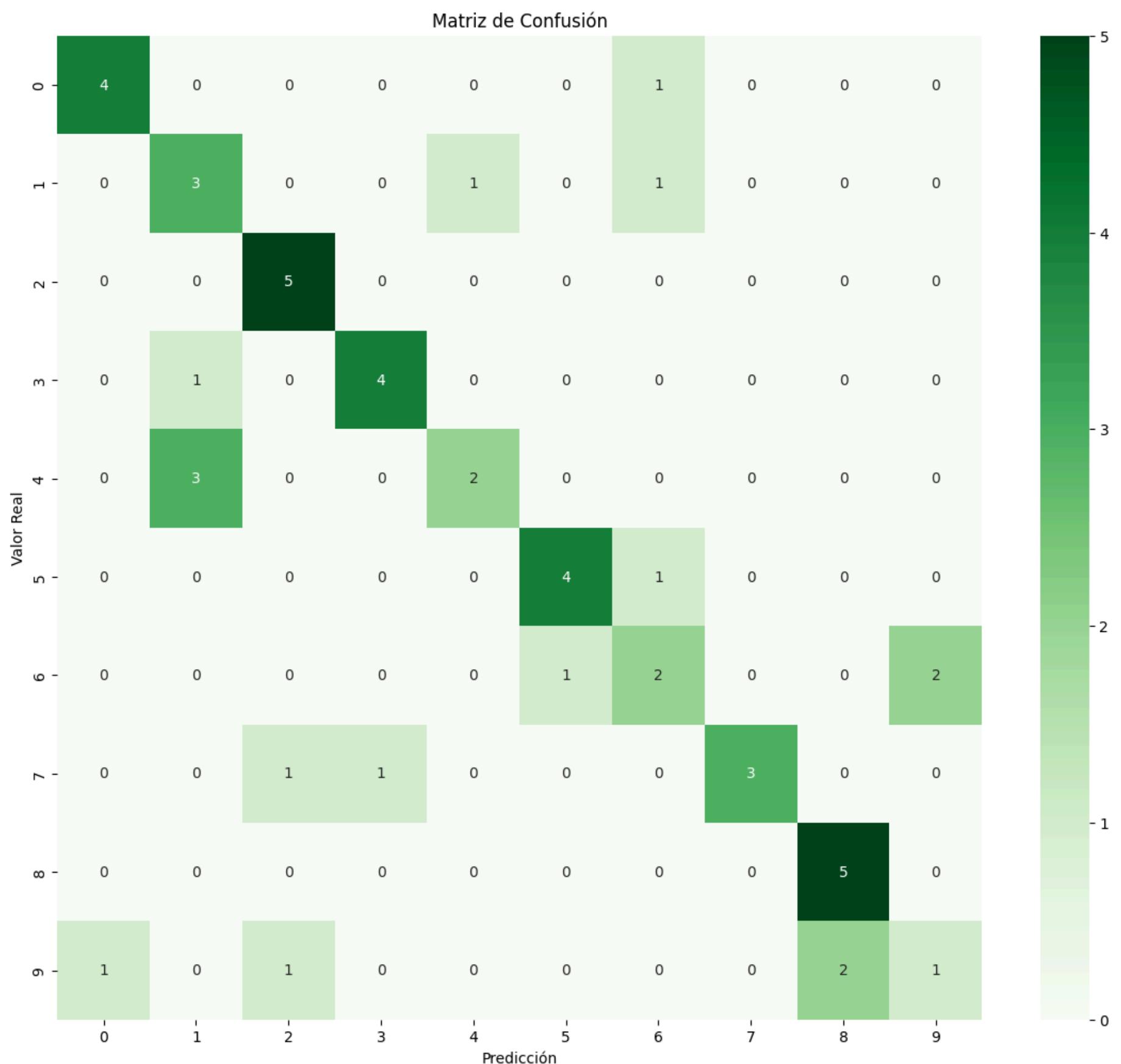
Predicción: 0, Etiqueta verdadera: 0



1/1 ————— 0s 21ms/step
Predicción: 0, Etiqueta verdadera: 0

Predicción: 0, Etiqueta verdadera: 0





Accuracy: 0.6600 (33/50)

Este código es el mismo que el de antes, sin embargo se realizaron cambios con ayuda de ChatGpt y Claude, primero se normalizaron los datos, esto porque favorece al entrenamiento de modelos, y como sabemos que las imágenes se procesan en RGB sabemos que se almacenan números enteros entre **0** y **255**, después se añadieron capas en nuestra red neuronal, un detalle importante es que se dice que al hacer una capa de red convolucional podemos obtener mejores resultados y por ultimo se realizaron cambios similares en la función que permite procesar las imágenes al formato en que se entreno la base de datos. Como se ve, se tardó más tiempo en entrenar el modelo, pero el accuracy mejoró tanto en los datos de prueba como con los de las imágenes que le cargamos. Este modelo se guardó en el repositorio para poder pasarlo a otro programa.

6. Considerando los cambios del punto anterior, genera un sistema que prediga en tiempo real los dígitos del **0** al **9** a partir de la imagen capturada por la cámara de la computadora, o algún otro dispositivo.

```
In [ ]: # Redes_Neuronales.py
```

Este es el nombre del archivo donde se ejecutó el programa en tiempo real, ese programa fue hecho en parte por mí aunque más que nada fue corregido por Claude.

Firma de Honor: Doy mi palabra que he realizado esta actividad con integridad académica