# PLAN DE TRABAJO DEL ESTUDIANTE

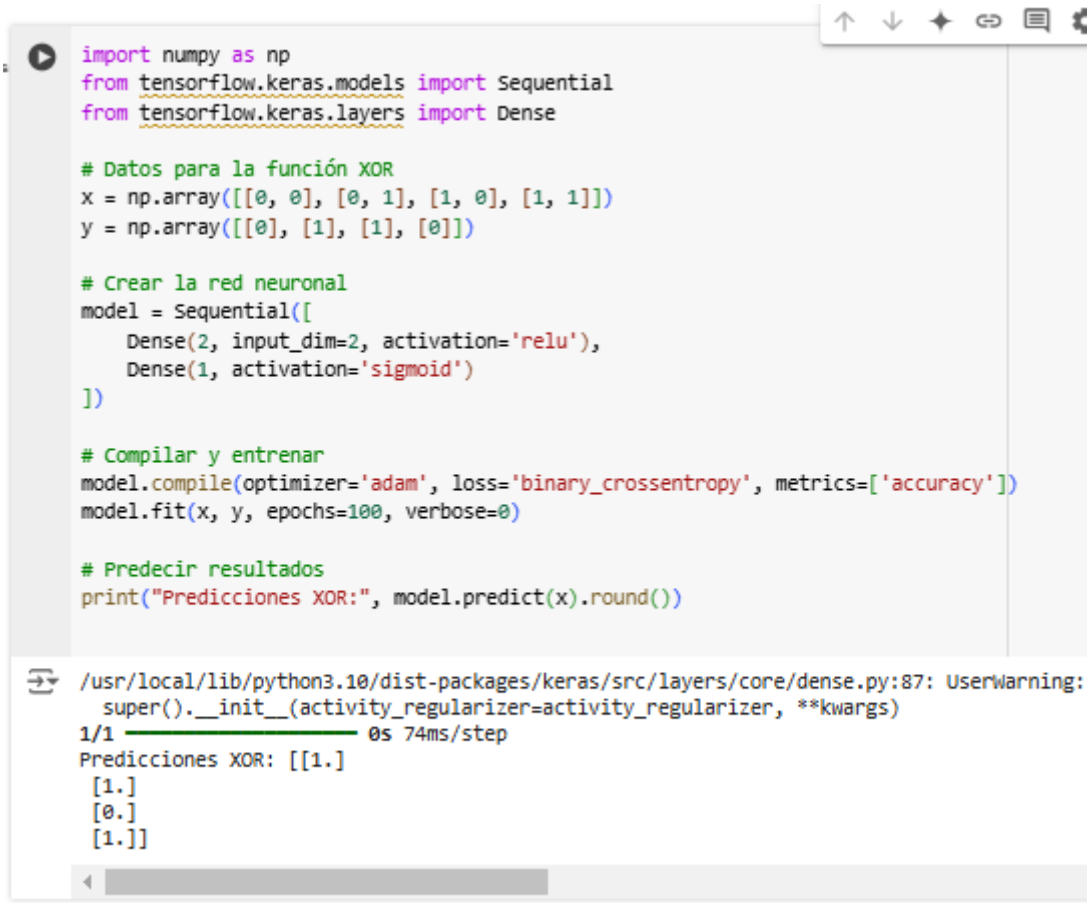
## 1. INFORMACIÓN GENERAL

| | | | |
|---|---|---|---|
| Apellidos y Nombres: | Gandy William Humiri Quispe | ID: | 1546329@senati.pe |
| Dirección Zonal/CFP: | Tacna/moquegua | | |
| Carrera: | Ingenieria de Software con Inteligencia Artificial | Semestre: | 4ciclo |
| Curso/ Mód. Formativo | MACHINE LEARNING Y DEEP LEARNING | | |
| Tema del Trabajo: | **Presentar informe primera parte del proyecto final** | | |

## 1: Define la red neuronal artificial y su importancia en la IA: Explica qué es una red neuronal artificial y por qué es importante en el campo de la inteligencia artificial.

Una red neuronal artificial (RNA) es un modelo computacional inspirado en el cerebro humano. Utiliza capas de nodos (neuronas) conectados para aprender patrones y relaciones complejas a partir de datos.

Ejemnplo: Las RNA son la base de deep Learning**,** permitiendo resolver tareas como visión por computadora, reconocimiento de voz y análisis de datos.



```python
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Datos para la función XOR
x = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([[0], [1], [1], [0]])

# Crear la red neuronal
model = Sequential([
    Dense(2, input_dim=2, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compilar y entrenar
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x, y, epochs=100, verbose=0)

# Predecir resultados
print("Predicciones XOR:", model.predict(x).round())
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
Predicciones XOR: [[1.]
 [1.]
 [0.]
 [1.]]
```

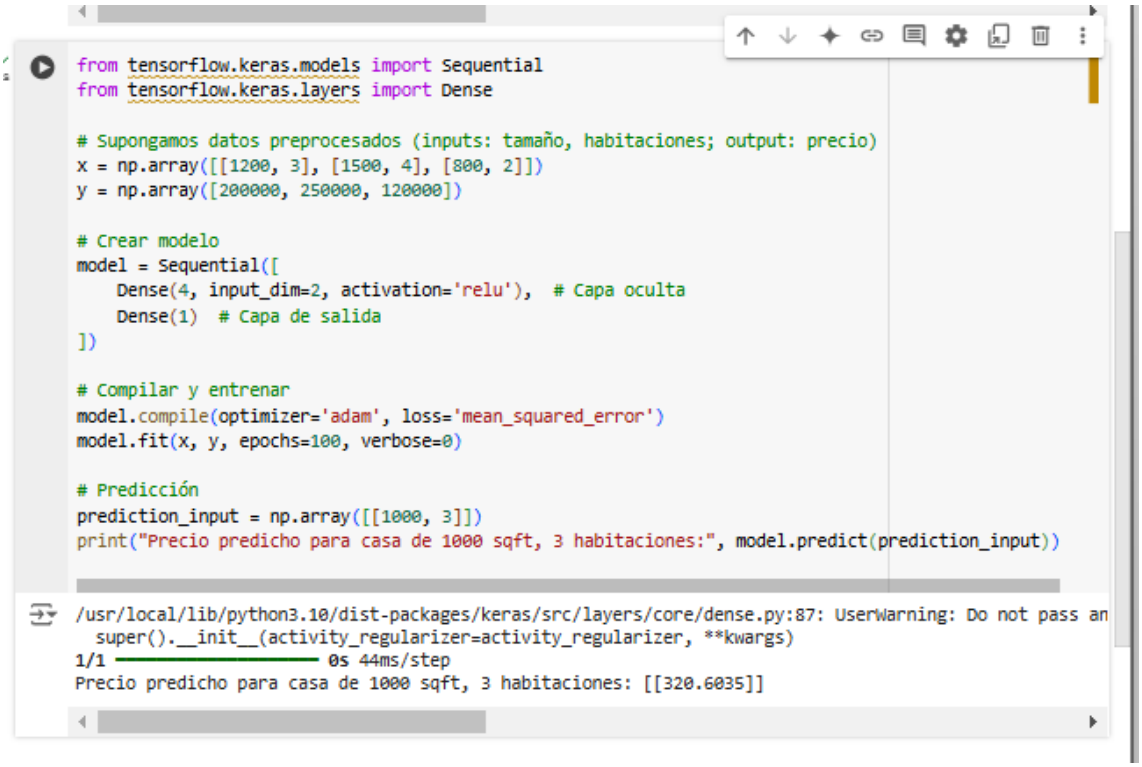
## 2: Describe la estructura de una red neuronal artificial: Explora la estructura básica de una red neuronal, incluyendo capas, neuronas y conexiones.

**Capa de entrada** Recibe las características del problema.

**Capas ocultas** Procesan información y su número y tamaño definen la capacidad del modelo.

**Capa de salida** Genera el resultado.

Cada conexión tiene un peso ajustado durante el entrenamiento.



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Supongamos datos preprocesados (inputs: tamaño, habitaciones; output: precio)
x = np.array([[1200, 3], [1500, 4], [800, 2]])
y = np.array([200000, 250000, 120000])

# Crear modelo
model = Sequential([
    Dense(4, input_dim=2, activation='relu'),  # Capa oculta
    Dense(1)  # Capa de salida
])

# Compilar y entrenar
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x, y, epochs=100, verbose=0)

# Predicción
prediction_input = np.array([[1000, 3]])
print("Precio predicho para casa de 1000 sqft, 3 habitaciones:", model.predict(prediction_input))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
Precio predicho para casa de 1000 sqft, 3 habitaciones: [[320.6035]]
```

## 3: Identifica los tipos de redes neuronales artificiales: Presenta diferentes tipos de redes neuronales, como redes neuronales convolucionales (CNN) y redes neuronales recurrentes (RNN).

1 **ANN (Red básica)** Clasificación o regresión general.
2 CNN **(Red convolucional)** Pocesa imágenes o datos espaciales
3 RNN **(Red recurrente)** Maneja datos secuenciales.

```python
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Cargar datos
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(-1, 28, 28, 1) / 255.0  # Normalización y reshape
x_test = x_test.reshape(-1, 28, 28, 1) / 255.0

# Crear modelo CNN
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

# Compilar y entrenar
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, batch_size=32, verbose=1)

# Evaluar modelo
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Precisión: {accuracy}")
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do n
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 46s 24ms/step - accuracy: 0.9145 - loss: 0.2943
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 41s 22ms/step - accuracy: 0.9831 - loss: 0.0553
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 82s 22ms/step - accuracy: 0.9894 - loss: 0.0325
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 81s 22ms/step - accuracy: 0.9937 - loss: 0.0204
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 41s 22ms/step - accuracy: 0.9959 - loss: 0.0129
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9793 - loss: 0.0625
Precisión: 0.9819999933242798
```

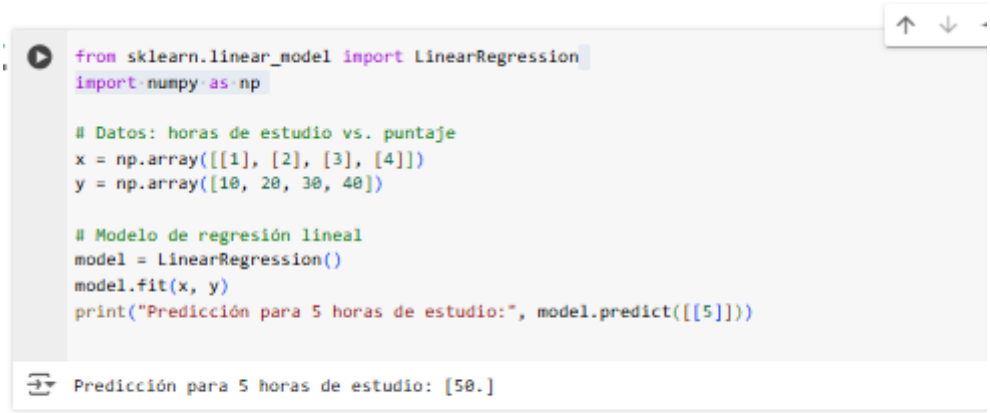
**4: Crea una red neuronal con Tensorflow y Keras: Desarrolla un modelo de red neuronal utilizando Tensorflow y Keras en Python para clasificar dígitos escritos a mano.**

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Cargar y preprocesar datos
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
y_train, y_test = to_categorical(y_train), to_categorical(y_test)

# Crear modelo
model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

# Compilar y entrenar
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)

# Evaluar
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Pérdida: {loss}, Precisión: {accuracy}")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pa
  super().__init__(**kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8783 - loss: 0.4296
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9669 - loss: 0.1139
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 3ms/step - accuracy: 0.9774 - loss: 0.0778
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.9820 - loss: 0.0567
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9875 - loss: 0.0411
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9711 - loss: 0.0917
Pérdida: 0.07731690257787704, Precisión: 0.9758999943733215
```

## 5: Define la red neuronal artificial y su importancia en la IA: Explica qué es una red Machine Learning Y Deep Learning neuronal artificial y por qué es importante en el campo de la inteligencia artificial.

**Machine Learning:** Métodos para que una máquina aprenda patrones (e.g., árboles de decisión).
**Deep Learning:** Subcampo que usa redes neuronales profundas para aprendizaje.

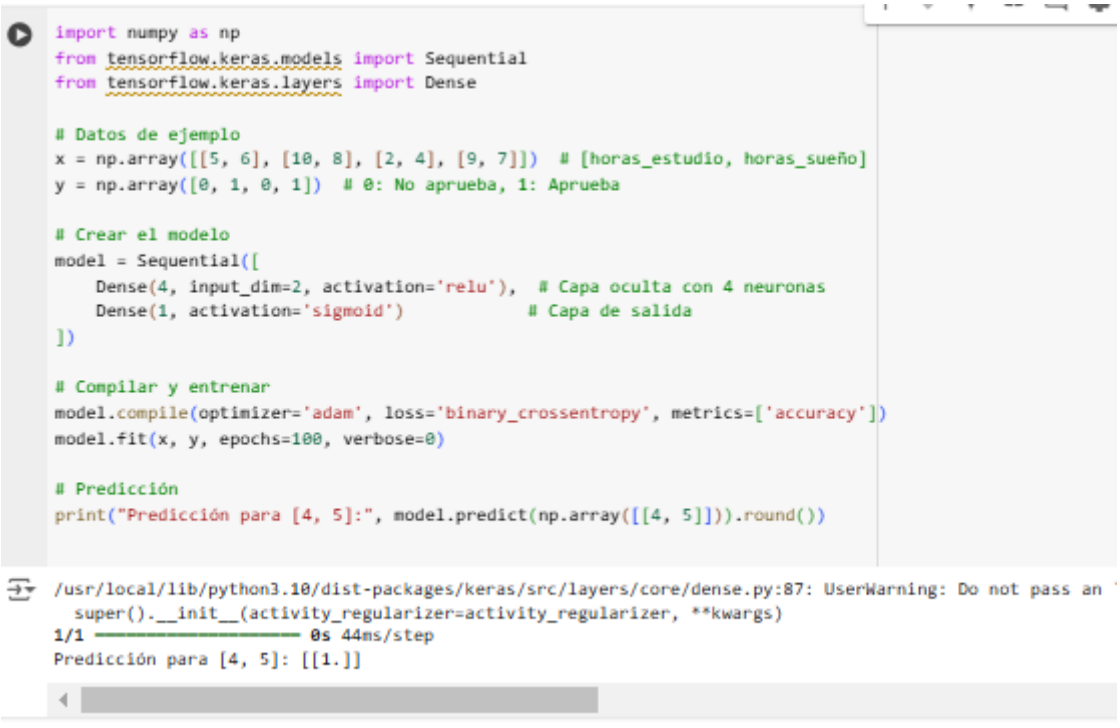


```
from sklearn.linear_model import LinearRegression
import numpy as np

# Datos: horas de estudio vs. puntaje
x = np.array([[1], [2], [3], [4]])
y = np.array([10, 20, 30, 40])

# Modelo de regresión lineal
model = LinearRegression()
model.fit(x, y)
print("Predicción para 5 horas de estudio:", model.predict([[5]]))
```

```
Predicción para 5 horas de estudio: [50.]
```

## 6: Describe la estructura de una red neuronal artificial: Explora la estructura básica de una red neuronal, incluyendo capas, neuronas y conexiones.

Una RNA tiene una organización jerárquica compuesta por capas y neuronas conectadas



```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Datos de ejemplo
x = np.array([[5, 6], [10, 8], [2, 4], [9, 7]])  # [horas_estudio, horas_sueño]
y = np.array([0, 1, 0, 1])  # 0: No aprueba, 1: Aprueba

# Crear el modelo
model = Sequential([
    Dense(4, input_dim=2, activation='relu'),  # Capa oculta con 4 neuronas
    Dense(1, activation='sigmoid')             # Capa de salida
])

# Compilar y entrenar
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x, y, epochs=100, verbose=0)

# Predicción
print("Predicción para [4, 5]:", model.predict(np.array([[4, 5]])).round())
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
Predicción para [4, 5]: [[1.]]
```

## 7: Identifica los tipos de redes neuronales artificiales: Presenta diferentes tipos de redes neuronales, como redes neuronales convolucionales (CNN) y redes neuronales recurrentes (RNN).

1: ANN (Redes neuronales artificiales)

2: CNN (Redes convolucionales)

3: RNN (Redes recurrentes)

4: GAN (Redes generativas adversarias)

```python
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Cargar datos
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = x_train.reshape(-1, 28, 28, 1) / 255.0
x_test = x_test.reshape(-1, 28, 28, 1) / 255.0

# Crear modelo CNN
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),  # Convolución
    MaxPooling2D((2, 2)),                                            # Submuestreo
    Flatten(),                                                       # Aplana
    Dense(128, activation='relu'),                                   # Capa oculta
    Dense(10, activation='softmax')                                  # Capa de salida
])

# Compilar y entrenar
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, batch_size=32, verbose=1)

# Evaluar
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Precisión en datos de prueba: {accuracy}")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 42s 22ms/step - accuracy: 0.9131 - loss: 0.2961
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 41s 22ms/step - accuracy: 0.9843 - loss: 0.0492
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 48s 26ms/step - accuracy: 0.9917 - loss: 0.0288
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 76s 22ms/step - accuracy: 0.9942 - loss: 0.0186
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 84s 23ms/step - accuracy: 0.9959 - loss: 0.0123
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9827 - loss: 0.0586
Precisión en datos de prueba: 0.9869999885559082
```

**8: Crea una red neuronal con Tensorflow y Keras: Desarrolla un modelo de red neuronal utilizando Tensorflow y Keras en Python para clasificar dígitos escritos a mano.**

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

# Cargar datos
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0  # Normalizar datos
y_train, y_test = to_categorical(y_train), to_categorical(y_test)

# Crear modelo
model = Sequential([
    Flatten(input_shape=(28, 28)),  # Aplana la imagen 28x28
    Dense(128, activation='relu'), # Capa oculta con 128 neuronas
    Dense(10, activation='softmax') # Capa de salida para 10 categorías (0-9)
])

# Compilar y entrenar
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, batch_size=32)

# Evaluar el modelo
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Pérdida: {loss}, Precisión: {accuracy}")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not
  super().__init__(**kwargs)
Epoch 1/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8797 - loss: 0.4237
Epoch 2/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.9641 - loss: 0.1241
Epoch 3/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9775 - loss: 0.0778
Epoch 4/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9826 - loss: 0.0574
Epoch 5/5
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.9858 - loss: 0.0452
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.9740 - loss: 0.0836
Pérdida: 0.07257243990898132, Precisión: 0.977400004863739
```