Universidad Sergio Arboleda

Facultad de Ciencias Exactas e Ingeniería

Proyecto Final Corte 2

Computación 2

Autor:

Mariana Rodríguez Pérez

Mayo 2025

Objetivo

Construir un modelo de red neuronal que clasifique imágenes de dígitos del 0 al 9, utilizando el conjunto de datos MNIST. Además, se pondrá a prueba la generalización del modelo con una imagen externa dibujada manualmente.

Descripción del modelo y arquitectura

Se utilizó una red neuronal con multiplles capas compuesta con una arquitectura compuesta de la siguiente forma:

- **Capa de entrada:** `Flatten`, que convierte una imagen 28x28 en un vector de 784 elementos.
- **Capa oculta:** `Dense` con 128 neuronas y activación ReLU.
- **Capa de salida:** `Dense` con 10 salidas, una por cada clase (dígito del 0 al 9), sin activación (logits).

Estrategia de entrenamiento

- **Dataset utilizado:** MNIST, con 60,000 imágenes de entrenamiento y 10,000 de prueba.
- **Optimización:** Algoritmo `Adam`.
- **Función de pérdida:** `SparseCategoricalCrossentropy` con `from_logits=True`.
- **Épocas:** 10.
- **Normalización previa de imágenes:** valores de píxeles entre 0 y 1.

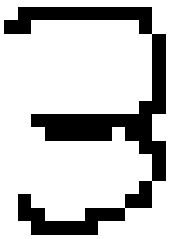
Resultados y análisis

Precisión del modelo

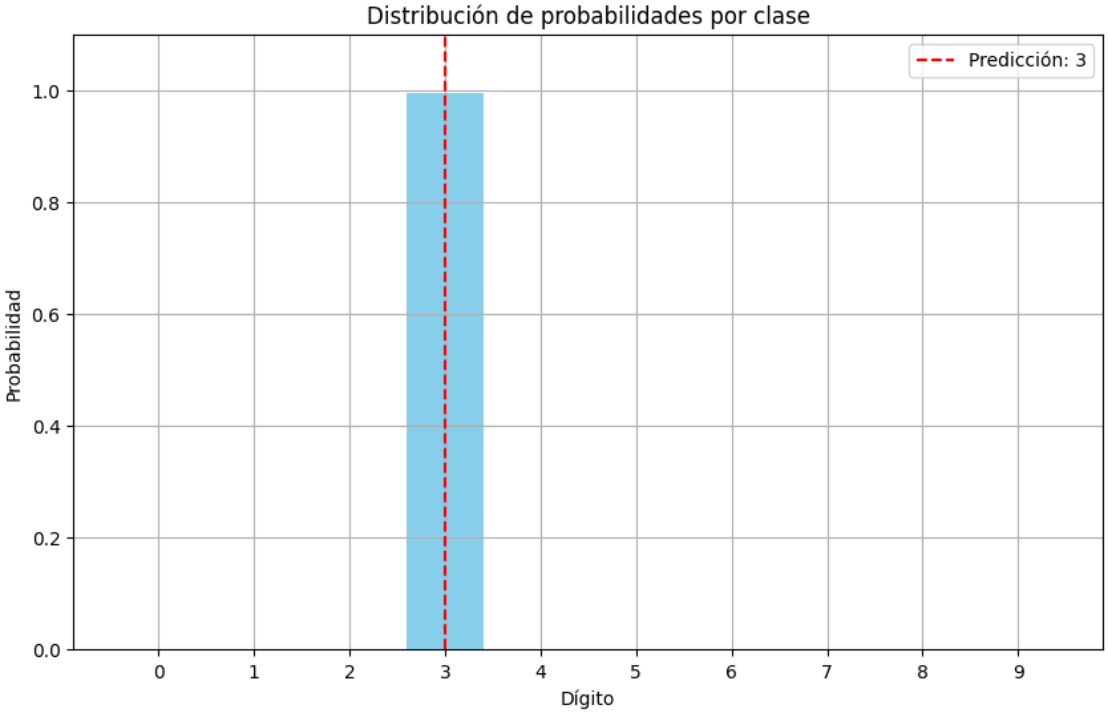
La precisión sobre el conjunto de datos de prueba fue : `0.9781`.

Predicción sobre imagen externa

Al modelo se le dio una imagen a analizar, la cual buscaba representar el número 3; (véase a continuación)



- Aplicando lo aprendido anteriormente, el modelo predijo el dígito: `3`.
- La distribución de probabilidades fue la siguiente:



Análisis

Se observa que el modelo logró identificar correctamente el número dibujado. Las clases más probables fueron coherentes con el contenido visual (Solo hubo una, puesto que el modelo fue muy preciso). Se detecta buen poder de generalización a pesar de haber entrenado con solo 10 épocas y una arquitectura simple.

Conclusiones

- La red neuronal implementada alcanzó una precisión adecuada con una arquitectura sencilla.
- El preprocesamiento adecuado (resizing, normalización e inversión de color) permitió que el modelo pudiera trabajar con una imagen externa.
- Como mejoras futuras, se pueden incluir capas adicionales, uso de regularización (Dropout) y entrenamiento por más épocas para optimizar la generalización.

Código y Proceso

A continuación adjunto el proceso en pdf realizado en google collab:

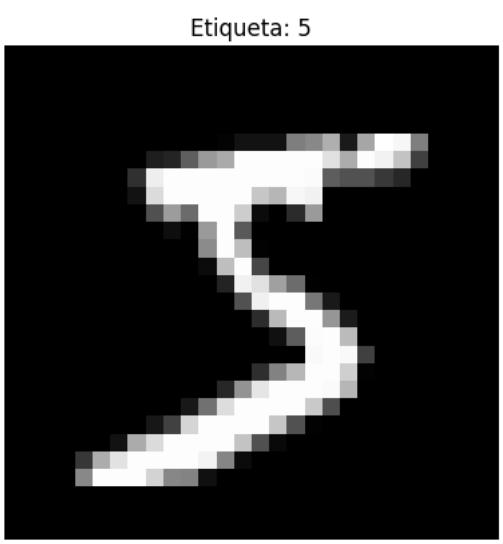
Proyecto Corte 2 – Computación Científica 2

Clasificación automática de dígitos manuscritos con redes neuronales

Objetivo: Construir un modelo de red neuronal que clasifique imágenes de dígitos (0–9) del dataset MNIST y lo pruebe con una imagen propia creada manualmente.

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing import image
from PIL import Image
```

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Este paso muestra una imagen del dataset
plt.imshow(x_train[0], cmap='gray')
plt.title(f"Etiqueta: {y_train[0]}")
plt.axis('off')
plt.show()
```




```
# Aquí se normalizan los valores entre 0 y 1
x_train = x_train / 255.0
x_test = x_test / 255.0
```

```
model = Sequential([
    layers.Flatten(input_shape=(28, 28)),
    layers.Dense(128, activation='relu'),
    layers.Dense(10)
])
```

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=10)
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8791 - loss: 0.4290
```

```
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 11s 3ms/step - accuracy: 0.9648 - loss: 0.1207
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9764 - loss: 0.0798
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.9818 - loss: 0.0584
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.9861 - loss: 0.0447
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.9900 - loss: 0.0324
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9909 - loss: 0.0283
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9943 - loss: 0.0200
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9951 - loss: 0.0158
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.9957 - loss: 0.0146
<keras.src.callbacks.history.History at 0x793c07db4e10>
```

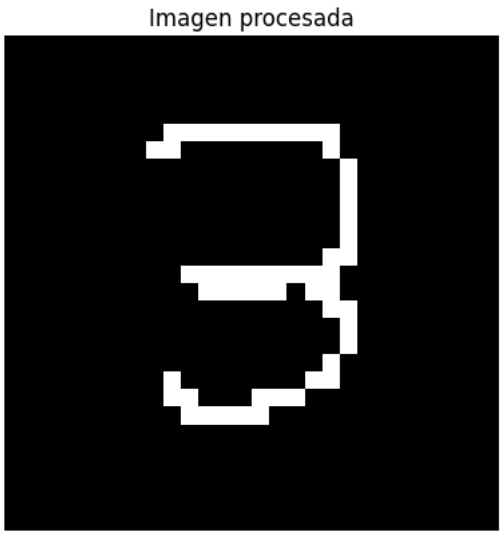
```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"Precisión en datos de prueba: {test_acc:.4f}")
```

```
313/313 - 1s - 3ms/step - accuracy: 0.9781 - loss: 0.0819
Precisión en datos de prueba: 0.9781
```

```
from google.colab import files
uploaded = files.upload()  # Espacio para subir la imagen a analizar
```

```
Elegir archivos Imagen proyecto1.png
• Imagen proyecto1.png(image/png) - 647 bytes, last modified: 1/5/2025 - 100% done
Saving Imagen proyecto1.png to Imagen proyecto1.png
```

```
def procesar_imagen_personal(ruta_imagen):
    """
    Carga y transforma una imagen en escala de grises, 28x28, normalizada.
    """
    imagen = Image.open(ruta_imagen).convert("L").resize((28, 28))
    imagen_np = np.array(imagen)

    # Invertir si el fondo es blanco y el número negro
    imagen_np = 255 - imagen_np

    # Mostrar imagen procesada
    plt.imshow(imagen_np, cmap='gray')
    plt.title("Imagen procesada")
    plt.axis('off')
    plt.show()

    # Normalizar
    imagen_np = imagen_np / 255.0
    imagen_np = imagen_np.reshape(1, 28, 28)

    return imagen_np



# Usar el primer archivo subido
nombre_archivo = next(iter(uploaded))
imagen_procesada = procesar_imagen_personal(nombre_archivo)

# Predicción
logits = model.predict(imagen_procesada)
prediccion = np.argmax(logits)
print(f"🔢 El modelo predice que el número es: {prediccion}")
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
El modelo predice que el número es: 3
```

```
# Aplicar softmax a los logits para obtener probabilidades
probabilidades = tf.nn.softmax(logits[0]).numpy()

# Gráfico de barras
plt.figure(figsize=(10, 6))
plt.bar(range(10), probabilidades, color='skyblue')
plt.xlabel('Dígito')
plt.ylabel('Probabilidad')
plt.title('Distribución de probabilidades por clase')
plt.xticks(range(10))
plt.ylim(0, 1.1)

# Marcar la predicción más probable
predicho = np.argmax(probabilidades)
plt.axvline(predicho, color='red', linestyle='--', label=f'Predicción: {predicho}')
plt.legend()
plt.grid(True)
plt.show()
```