Informe

Proyecto: Clasificación y predicción de dígitos escritos a mano usando redes neuronales en Python
Autor: Juan Sebastián López Galvis
Fecha: Mayo 02 de 2025

---

1. Introducción

Este proyecto utiliza redes neuronales para clasificar imágenes de los dígitos entre 0 y 9, entrenando un modelo con el dataset MNIST de Tensorflow. Además, se evalúa la capacidad del modelo ya entrenado y puesto a prueba para generalizar las predicciones usando imágenes creadas a mano.

---

2. Objetivo

Desarrollar un modelo de que identifique y clasifique correctamente números escritos a mano, tanto de un conjunto estándar como de una imagen personalizada hecha a mano, mediante el uso de redes neuronales.

---

3. Herramientas utilizadas

- **Lenguaje**: Python
- **Entorno**: Google Colab / Jupyter Notebook
- **Librerías**: TensorFlow, Keras, NumPy, Matplotlib

---

4. Dataset utilizado

Se utilizó el conjunto de datos Mnist, que contiene 70,000 imágenes repartidas en 60,000 para entrenamiento y 10,000 para prueba (esto es por defecto, aunque puede personalizarse), imágenes en escala de grises de tamaño 28x28 píxeles ya que por defecto ese es el tamaño que utiliza el dataset mnisit, Cada imagen representa un dígito del 0 al 9.

---

5. Arquitectura del modelo

Se construyó una red neuronal con las siguiente arquitectura:

```
# arquitectura de la red neuronal
modelo = keras.Sequential([
    layers.Flatten(input_shape=(28, 28, 1)),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

- Se utiliza **ReLU** para acelerar el aprendizaje y evitar saturaciones.
- La capa de salida utiliza **Softmax** para clasificar entre 10 clases (0 a 9).

---

6. Entrenamiento

En la variable 'historial' aplicamos la función .fit al modelo para entrenarlo. Para este caso elegimos cinco épocas de entrenamiento y esto nos arroja en la prueba una precisión de más de 97%, lo que nos indica un rendimiento excelente del modelo.

```
# Entrenamos el modelo
historial = modelo.fit(datos_entrenamiento, etiquetas_entrenamiento,
                       epochs=5,
                       validation_data=(datos_prueba, etiquetas_prueba))
```

```
# probamos el modelo, lo evaluamos
puntaje = modelo.evaluate(datos_prueba, etiquetas_prueba)
print(f"Precisión del modelo: {puntaje[1]:.4f}")
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9697 - loss: 0.0956
Precisión del modelo: 0.9741
```

---

7. Clasificación de imagen externa

Dibujamos un número usando un programa externo (Paint), lo guardamos como imagen .jpg la adaptamos para que coincida con el formato de entrada que el modelo recibe. Esto es:

- **Adaptación aplicada**:
  - Escala de grises
  - Redimensionado a 28x28
  - Inversión de colores (si el fondo era blanco) aunque este paso es opcional.
  - Normalización de píxeles (división entre 255)

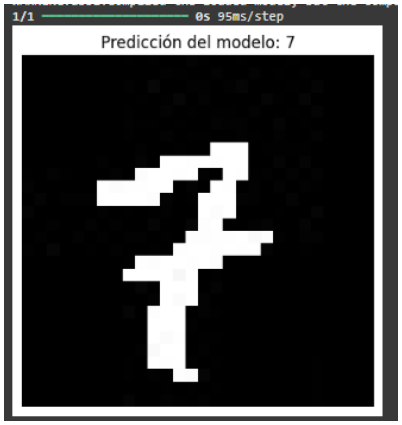
- Aplanado a vector de 784 elementos

Ejemplo de predicción externa:

---

8. Análisis

El modelo funciona muy bien prediciendo y clasificando imágenes con datos similares a las proporcionadas por el tataset de Mnisit durante el entrenamiento. Sin embargo, para imágenes propias, el rendimiento y precisión aun cuando es muy bueno, puede depender mucho del contraste, la calidad de la imagen, distorsión o baja resolución además de imágenes semejantes que el modelo puede confundir.



---

9. Conclusiones

El proyecto demuestra cómo una red neuronal puede aprender a identificar patrones visuales complejos, como números escritos a mano. El sistema tiene un alto rendimiento en datos similares a los de entrenamiento, y se puede conseguir un mejor rendimiento para imágenes externas nuevas aplicándoles el proceso adecuado.