Universidad Autónoma de Estado de México

Centro Universitario UAEM Zumpango

Ingeniería en Computación

Ciclo escolar 2022-B

Redes Neuronales

Dr. Asdrubal López Chao

“Proyecto Identificación de números (Dígitos) a través de imágenes.”

Presenta:

López Mendoza Mauricio Gabino
Vilchis Medina Luis Julian

Introducción.

El OCR o Reconocimiento Óptico de Caracteres es la identificación de caracteres impresos mediante el uso de dispositivos fotoeléctricos y software. Si bien el trabajo con los OCR no es nuevo y existen varios métodos para la clasificación de los patrones existen aún rutas que resolver para llegar a poder aplicarlos con mayor eficacia. Algunos de los métodos tradicionalmente usados se basan en los algoritmos de k-vecinos, las maquinas de soporte vectorial (SVM) y comparación contra plantillas (Template Matching).
El interés de utilizar las redes neuronales en el OCR recientemente ha sido debido a su potencial considerando la manipulación libre de topología que la integrará, y que varios trabajos.

La motivación para la elaboración del presente trabajo es el interés sobre la versatilidad de las redes neuronales convencionales en una gran diversidad de ámbitos y sobre su capacidad para resolver problemas de alta complejidad sin necesidad de conocer su solución, con el objetivo de implementar una red neuronal que se podrá adaptar a otros proyectos en el futuro.

Utilizaremos Tensor Flow 2 junto a Keras para programar una Red Neuronal Convolucional (CNN) destinada a reconocer imágenes de números. Para esto vamos a disponer de un famoso dataset denominado MNIST y llamado a veces el "Hello world" de las Redes Neuronales Artificiales. Este dataset está separado en 60000 imágenes de entrenamiento y 10000 para el test, los cuales pueden ser obtenidos desde el tf.keras.

Objetivos

Objetivos: Este proyecto busca desarrollar y evaluar una red neuronal en el software Google colab e implementarla para obtener los mejores resultados, ya que ayudará a reconocer los dígitos del número de identificación de manera más rápida y efectiva.

- Dividir las imágenes digitalizadas para extraer cada número.
- Leer la imagen de cada dígito y tratar la imagen .
- Definir los métodos para obtener características de las imágenes.
- Obtención de las características de las imágenes.
- Buscar el mejor resultado óptimo.

Bibliotecas utilizadas

- Tensorflow
- Pandas

- Matplotlib
- Numpy

Código.

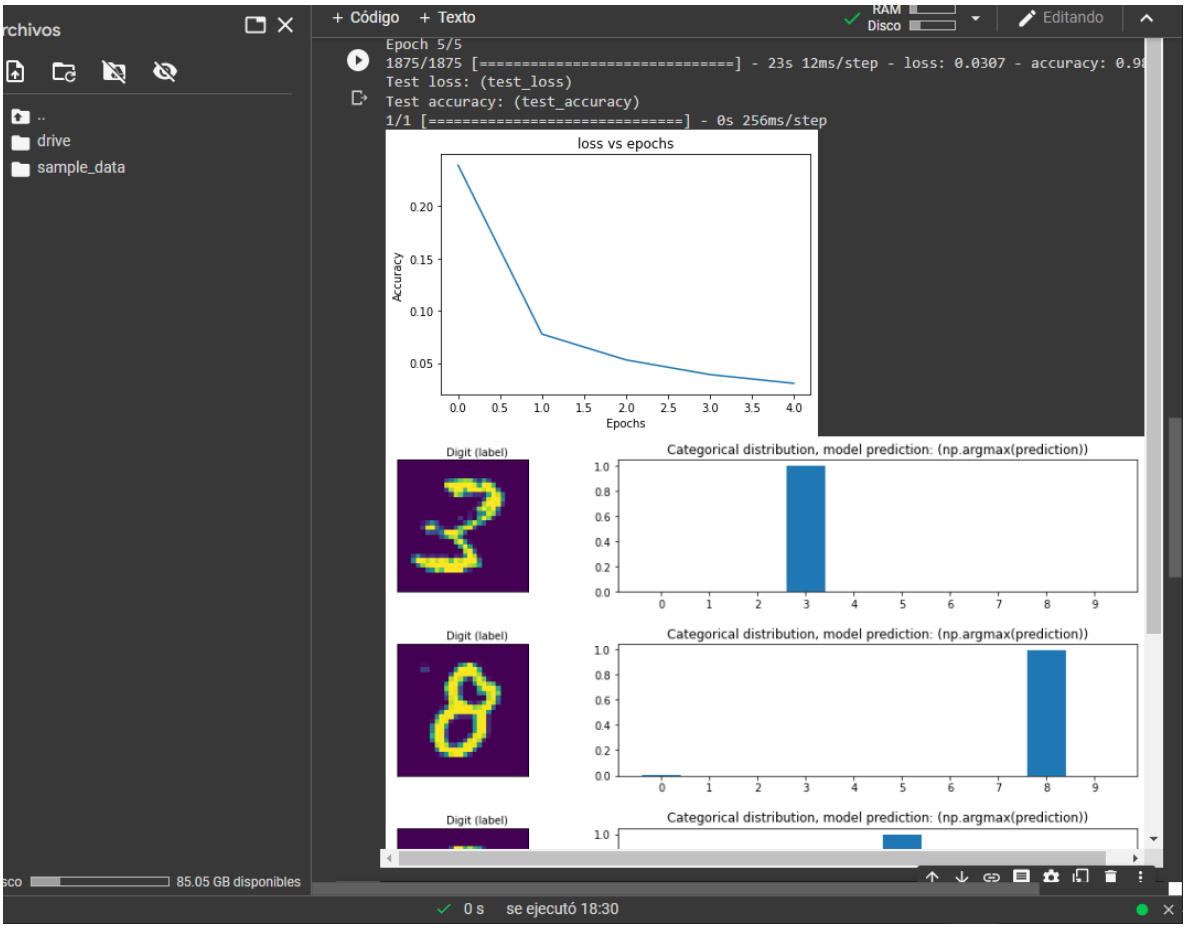
```
"""
UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO
CU UAEM ZUMPANGO
UA: Redes Neuronales
Tema: Red Neuronal con Tensorflow y MNIST
Alumnos:
      Lopez Mendoza Mauricio Gabino
      Vilchis Medina Luis Julian

Profesor: Dr. Asdrúbal López Chau
Descripción: Red Neuronal destinada al reconocimiento de digitos
"""
#Bibliotecas a utilizar, tensorflow y keras para el reconocimiento y entrenamiento
#pandas para leer archivos de nuestro drive desde colab
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

#Cargamos con ayuda de nuestro dataset MNIST que es el que nos ayudara a
entrenar
#nuestra red neuronal
mnist_data = tf.keras.datasets.mnist
#Muestras del entrenamiento
(train_images, train_labels), (test_images, test_labels) = mnist_data.load_data()
train_images_scaled = train_images/255
test_images_scaled = test_images/255
np.amin(train_images_scaled)
train_images_scaled = train_images_scaled[...,np.newaxis]
test_images_scaled = test_images_scaled[...,np.newaxis]
train_images_scaled.shape
#Construimos el modelo con una red neuronal secuencial "softmax"
model =
Sequential([Conv2D(8,(3,3),activation='relu',padding='SAME',input_shape=train_im
ages_scaled[0].shape),
                MaxPooling2D((2,2)),
                Flatten(),
                Dense(64, activation='relu'),
                Dense(64, activation='relu'),
```

```
                Dense(10, activation='softmax')
                ])
model.summary()
#Funcion de perdida y optimizador adam
train_images_scaled[0].shape
model.compile(optimizer='adam',
            loss = 'sparse_categorical_crossentropy',
            metrics=['accuracy'])
#Entrenamiento y numero de epocas para mas precision en este caso utilizamos
15
#Numero de iteraciones y precision que nos dara
history = model.fit(train_images_scaled, train_labels, epochs=15, verbose=1)

#Analisis de epocas, como es que va el testeo y lo mostramos a traves de una
grafica
frame = pd.DataFrame(history.history)

acc_plot = frame.plot(y="loss", title="loss vs epochs", legend=False)
acc_plot.set(xlabel = "Epochs", ylabel = "Accuracy")

test_loss, test_accuracy = model.evaluate(test_images_scaled,test_labels,
verbose = 0)
print(f"Test loss: (test_loss)")
print(f"Test accuracy: (test_accuracy)")

#Con ayuda de nuestra dataset probamos nuestra red con diversos numeros y
hacemos las predicciones de este
num_test_images = test_images_scaled.shape[0]

random_inx = np.random.choice(num_test_images, 4)
random_test_images = test_images_scaled[random_inx,...]
random_test_labels = test_labels[random_inx, ...]

predictions = model.predict(random_test_images)

fig, axes = plt.subplots(4, 2, figsize=(16, 12))
fig.subplots_adjust(hspace=0.4, wspace = -0.2)

for i, (prediction, image, label) in enumerate(zip(predictions, random_test_images,
random_test_labels)):
  axes[i, 0].imshow(np.squeeze(image))
  axes[i, 0].get_xaxis().set_visible(False)
  axes[i, 0].get_yaxis().set_visible(False)
  axes[i, 0].text(10, -1.5, f'Digit (label)')
  axes[i, 1].bar(np.arange(len(prediction)), prediction)
  axes[i, 1].set_xticks(np.arange(len(prediction)))
  axes[i, 1].set_title(f"Categorical distribution, model prediction:
```

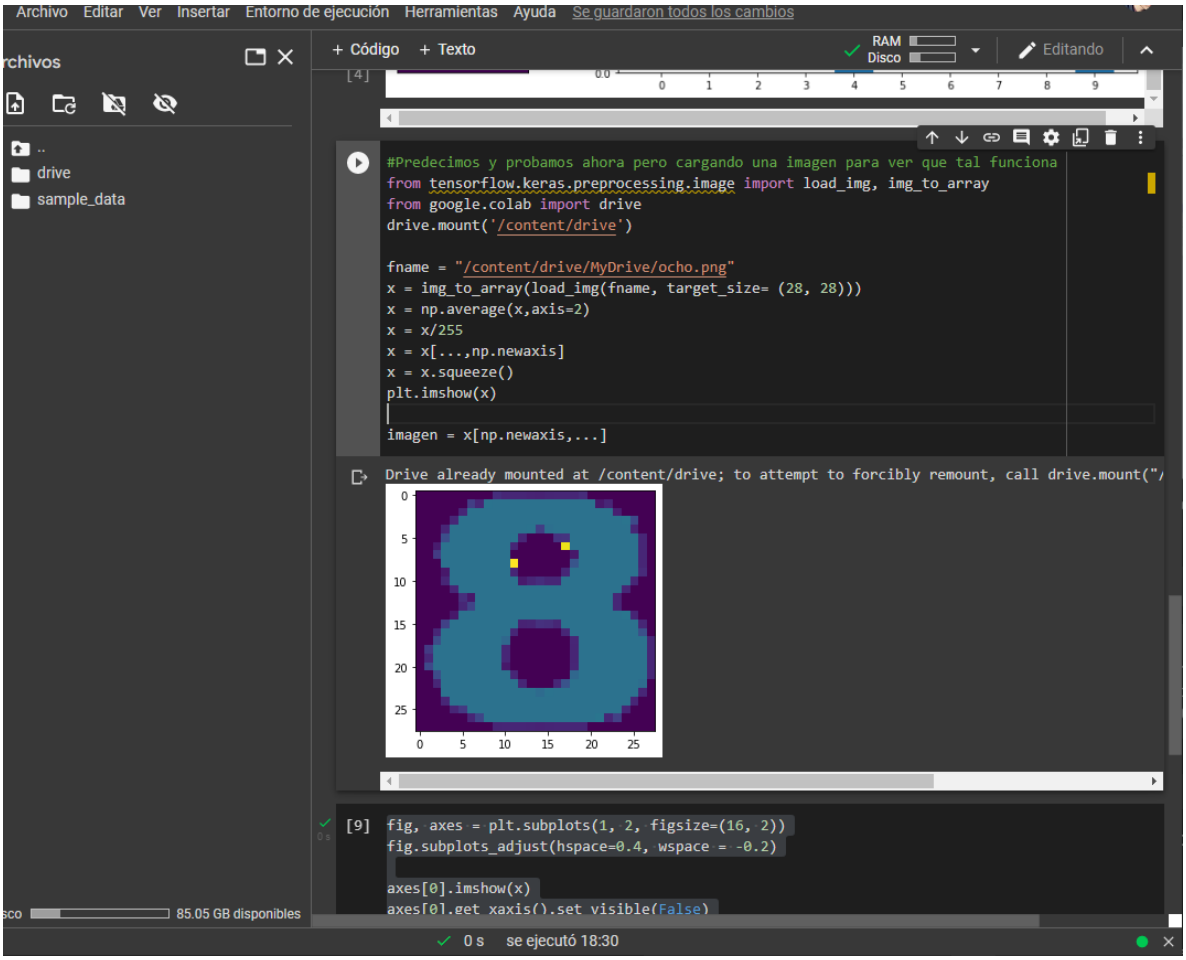
```
(np.argmax(prediction))")


#Predecimos y probamos ahora pero cargando una imagen para ver que tal funciona
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from google.colab import drive
drive.mount('/content/drive')

fname = "/content/drive/MyDrive/ocho.png"
x = img_to_array(load_img(fname, target_size= (28, 28)))
x = np.average(x,axis=2)
x = x/255
x = x[...,np.newaxis]
x = x.squeeze()
plt.imshow(x)

imagen = x[np.newaxis,...]

fig, axes = plt.subplots(1, 2, figsize=(16, 2))
fig.subplots_adjust(hspace=0.4, wspace = -0.2)

axes[0].imshow(x)
axes[0].get_xaxis().set_visible(False)
axes[0].get_yaxis().set_visible(False)
axes[0].text(10, -1.5,f'{fname}')
axes[1].bar(np.arange(len(prediction)), prediction)
axes[1].set_xticks(np.arange(len(prediction)))
axes[1].set_title(f"Prediccion del modelo: {np.argmax(prediction)}")
plt.show()
```

Desarrollo

- Entrenamiento


+ Código + Texto
RAM
Disco
Editando
Epoch 5/5
1875/1875 [==========] - 23s 12ms/step - loss: 0.0307 - accuracy: 0.9
Test loss: (test_loss)
Test accuracy: (test_accuracy)
1/1 [==========] - 0s 256ms/step
loss vs epochs
Accuracy
Epochs
Digit (label)
Categorical distribution, model prediction: (np.argmax(prediction))
Digit (label)
Categorical distribution, model prediction: (np.argmax(prediction))
Digit (label)
Categorical distribution, model prediction: (np.argmax(prediction))
rchivos
drive
sample_data
85.05 GB disponibles
0 s se ejecutó 18:30

- Prediccion



Conclusión.

El reconocimiento de dígitos es un entorno en creciente uso y por consiguiente requiere ir abordando alternativas para su implementación, el uso de redes neuronales ha venido retomando el auge dentro del área de reconocimiento de patrones. Este documento muestra el uso de redes neuronales, a través de un software personalizado, como el motor detrás un sistema de reconocimiento de caracteres ópticos. En este sistema los dígitos numéricos son simplificados a través de filtros de imagen y luego presentados como entrada a la red neuronal para entrenarla (usando el algoritmo de retro-propagación) y ser capaz de clasificar otras muestras en la etapa de pruebas. Los resultados muestran tasas de reconocimiento cercanas al 95%, que se pueden considerar como aceptables para topologías de una sola capa, dejando pendiente para futuros experimentos el trabajo con redes multicapa pre-entrenadas, ya que suelen incrementar fuertemente su eficiencia.