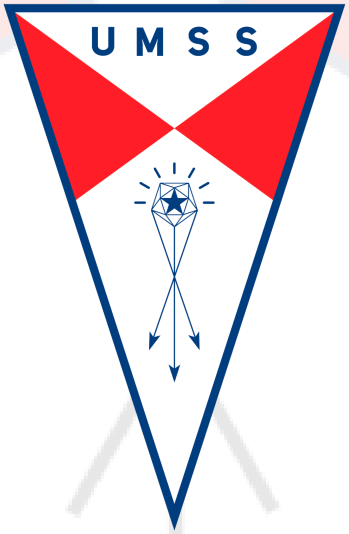
Trabajo práctico Primer Parcial Ciencia de Datos y Machine Learning



UNIVERSIDAD
MAYOR DE SAN SIMÓN
Ciencia y Conocimiento desde 1832

Integrantes:
Arraya Guzmán Eduardo Josué
Hinojosa Espinoza Alvaro
Herbas Chilo Rene Mijail

Docente: Lic. Erika Patricia Rodríguez Bilbao
Fecha de entrega: 06/05/2022

Cochabamba - Bolivia
2022

I. Introducción

La inteligencia artificial es una rama de la ingeniería informática en la cual se estudia las técnicas que se utilizan para dar la capacidad a una máquina de que piense y razone por su cuenta, es decir que las máquinas puedan imitar la inteligencia humana al momento de realizar las tareas y puedan mejorar su rendimiento con el paso del tiempo. El comienzo de la inteligencia artificial se dió gracias al surgimiento de una red neuronal simple modelando la misma funcionalidad de una neurona del cerebro mediante circuitos electrónicos.

Luego comenzó a tomar más importancia el aprendizaje automático (machine learning) en el cual se podía realizar una automatización para la construcción de modelos analiticos, es decir que los sistemas en base a unos datos de entrada, puedan reconocer por sí mismos, los diferentes patrones de los datos y a partir de estos realizar una decisión o una predicción a un dato externo a los datos de entrada, en otras palabras se quería que las máquinas aprendan sin la necesidad de programarlas para realizar esas tareas. La importancia del aprendizaje automático fue aumentando gracias al aumento del volumen de datos disponibles, cada vez eran más extensos y más complicados, por la necesidad de un procesamiento computacional más económico y poderoso, y por tener almacenamiento de datos.

Finalmente se originó el deep learning debido a que el machine learning tenía deficiencias en muchas tareas en cuanto a predicciones o reconocimientos, ya que esta se dedicaba directamente a reconocer los patrones específicos de los datos de entrada, en cambio el deep learning, ya no permitía la organización de datos en base a ecuaciones predefinidas, el deep learning permite el aprendizaje mediante la configuración de parámetros básicos para los datos de entrada y así entrenar a la máquina en muchas capas ocultas. Las áreas en donde el deep learning tiene más importancia, es en el reconocimiento de voz o en el reconocimiento de imágenes, como también para el procesamiento del lenguaje natural y para los sistemas de recomendación.

Para el trabajo utilizaremos las redes neuronales convolucionales (CNN) las cuales son un tipo de red neuronal artificial que utiliza aprendizaje supervisado, en la cual se imita la función del córtex visual del ojo humano para lograr identificar patrones o características en los datos de entrada, en este caso la red neuronal convolucional tiene consigo 2 capas ocultas especiales adicionales a las que ya se conocen de las redes neuronales (1 capa de entrada, 1 capa oculta y 1 capa de salida) la primera es una capa de convolución que le permite tomar grupos de píxeles en toda la imagen para poder identificar los píxeles que son más relevantes para la predicción, para eso se puede utilizar diferentes matrices llamadas kernel, y luego se encuentra

la capa de agrupación, la cual se encarga de agrupar las diferentes características que se consiguió en el kernel y de esta manera reducir la imagen y permitir encontrar características más relevantes.

II. Objetivos

- Permitir el reconocimiento de imágenes a partir de un dataset, utilizando redes neuronales convolucionales.

III. Marco Teórico

- **TensorFlow.-** Es una librería de código abierto que es utilizada para la computación numérica y el machine learning a gran escala, esta tiene un gran conjunto de modelos y algoritmos para machine learning y deep learning, es decir permite el entrenamiento y ejecución de diferentes redes neuronales.
- **Keras.-** Es una API de deep learning en Python que se ejecuta en base al aprendizaje automático, es simple para reducir la carga de trabajo del desarrollador y pueda concentrarse en las partes de problema que realmente importan, es flexible lo que permite flujos de trabajo simples o complejos, y es poderosa en cuanto al rendimiento y la escalabilidad.
- **Matplotlib.-** Es una librería que es utilizada para crear visualizaciones estáticas, animadas e interactivas en Python.
- **Numpy.-** Es una librería de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados y una variedad de rutinas para operaciones rápidas en matrices, que incluyen manipulación matemática, lógica, de formas, clasificación, etc.
- **Math.-** Es una librería que ofrece funciones matemáticas para uso en el campo de los números reales.

IV. Ingeniería

Actualmente existe el área del reconocimiento de imágenes, en la cual se requiere que un sistema tenga la capacidad de identificar cualquier imagen y dar una predicción acerca de lo que cree que es la imagen, es decir si puede ser una persona, un animal o cualquier otra cosa. Para nuestro caso tenemos la necesidad de entrenar una red neuronal que pueda identificar la imagen de un número e indicar qué número es el que se encuentra en la imagen, para lo

cual utilizamos la librería de tensorflow la cual nos permite poder cargar el dataset de MNIST a partir de Keras, además de que nos permite generar nuestro modelo con una red neuronal convolucional con 2 capas de convolución y otras 2 de agrupación, estas capas de convolución las configuramos con la función de activación ReLu que nos permite transformar los valores introducidos anulando valores negativos y dejando los positivos tal y como entran. De esta manera separamos los datos para los datos de entrenamiento y para los datos de pruebas y de esta forma entramos a entrenar el modelo.

a) Limpieza de datos

Al comenzar cargamos los datos en las distintas En cuanto a la limpieza de datos se recurrió a la normalización de los píxeles ya que el rango se encontraba entre 0-255 píxeles

```python
import tensorflow as tf
import tensorflow_datasets as tfds

#Descargar set de datos de MNIST (Numeros escritos a mano, etiquetados)
datos, metadatos = tfds.load('mnist', as_supervised=True, with_info=True)

#Obtener en variables separadas los datos de entrenamiento (600000) y pruebas (100000)
datos_entrenamiento, datos_pruebas = datos['train'], datos['test']
#Funcion de normalizacion para los datos (Pasar valor de los pixeles de 0-255 a 0-1)
#La red aprende mejor y mas rapido
def normalizar(imagenes, etiquetas):
  imagenes = tf.cast(imagenes, tf.float32)
  imagenes /= 255 #Aqui se pasa de 0-255 a 0-1
  return imagenes, etiquetas

#Normalizar los datos de entrenamiento con la funcion que hicimos
datos_entrenamiento = datos_entrenamiento.map(normalizar)
datos_pruebas = datos_pruebas.map(normalizar)

#Agregar a cache (usar memoria en lugar de disco, entrenamiento mas rapido)
datos_entrenamiento = datos_entrenamiento.cache()
datos_pruebas = datos_pruebas.cache()

clases = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
```

De esta forma se cambia el tipo de datos para cada una de las imágenes y se reescala sus píxeles a un rango de 0-1 para mejor manipulación de datos

b) Determinación de datos entrenamiento del modelo

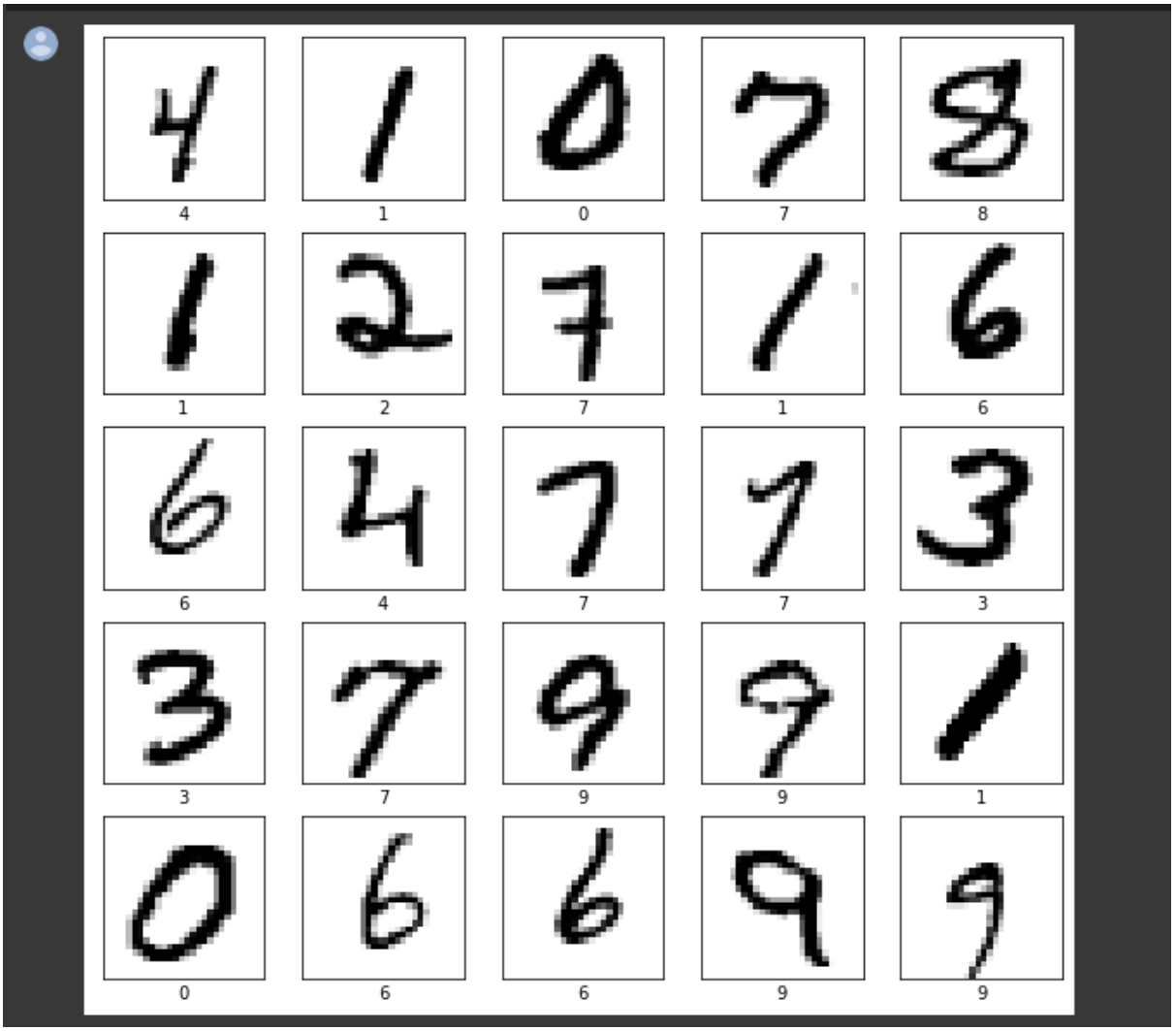
Se imprimen parte de los datos del dataset

```
#Codigo para mostrar imagenes del set
import matplotlib.pyplot as plt

plt.figure(figsize=(10,10))

for i, (imagen, etiqueta) in enumerate(datos_entrenamiento.take(25)):
  imagen = imagen.numpy().reshape((28,28))
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.grid(False)
  plt.imshow(imagen, cmap=plt.cm.binary)
  plt.xlabel(clases[etiqueta])

plt.show()
```



c) Determinación del modelo

Como anteriormente se mencionó utilizamos un modelo con 2 capas convolucionales y 2 capas de agrupación, luego se encuentra la red neuronal simple en la cual para la capa final utilizamos la función de activación softmax que calcula las probabilidades relativas, es decir devuelve las probabilidades para cada una de las clases del modelo, en nuestro caso las clases son del 1-9.

```
#Crear el modelo
#Cuenta con 1 capa de convolución con 32 núcleos y otra con 64. 2 capas de agrupación.
#Finalmente una capa densa con 100 neuronas
modelo = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), input_shape=(28,28,1), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2), #2,2 es el tamano de la matriz

    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2), #2,2 es el tamano de la matriz

    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=100, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

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

```
[33] #Los numeros de datos de entrenamiento y pruebas (60000 y 10000)
num_datos_entrenamiento = metadatos.splits["train"].num_examples
num_datos_pruebas = metadatos.splits["test"].num_examples

#Trabajar por lotes
TAMANO_LOTE=32

#Mezclar los datos de entrenamiento para que el modelo no aprenda el orden
datos_entrenamiento = datos_entrenamiento.repeat().shuffle(num_datos_entrenamiento).batch(TAMANO_LOTE)
datos_pruebas = datos_pruebas.batch(TAMANO_LOTE)
```

```
[34] #Realizar el entrenamiento
import math

historial = modelo.fit(
    datos_entrenamiento,
    epochs=60,
    steps_per_epoch=math.ceil(num_datos_entrenamiento/TAMANO_LOTE)
)
```

d) Validación con al menos 4 parámetros

```
score = modelo.evaluate(datos_pruebas, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 0.0935392677783966
Test accuracy: 0.991599977016449
```

V. Conclusiones

Al finalizar el trabajo, pudimos observar que las redes neuronales convolucionales hacen un gran trabajo para el reconocimiento de imágenes, por su buena capacidad de abstraer e identificar las diferentes características y patrones que pueda tener una imagen, mediante más capas convolucionales y más capas de agrupación se pueden ir consiguiendo patrones más específicos como ser líneas y curvas, que para el reconocimiento de imágenes es muy útil.

VI. Bibliografía

- https://www.sas.com/es_cl/insights/analytics/what-is-artificial-intelligence.html
- https://www.sas.com/es_cl/insights/analytics/deep-learning.html
- https://www.sas.com/es_cl/insights/analytics/machine-learning.html
- https://www.sas.com/es_cl/insights/analytics/neural-networks.html
- https://aprendeia.com/que-es-tensorflow-como-funciona/
- https://www.tensorflow.org/?hl=es-419
- https://keras.io/about/
- https://matplotlib.org/
- https://numpy.org/doc/stable/