

“Año del Bicentenario del Perú: 200 años de Independencia”

UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ



**FACULTAD DE
INGENIERIA DE SISTEMAS**



EVALUACIÓN DE PRIMER CONSOLIDADO

ASIGNATURA:

INGENIERIA DEL CONOCIMIENTO

CATÉDRATICO:

Dr. Abraham E. Gamarra Moreno

INTEGRANTES:

**CUICAPUZA MONTES, José Carlos
FABIAN SINTI, Estefany Sadith
MEDRANO QUISPE, Samuel
MUÑOZ INGARUCA, Piero Carlos
ROCA HORMAZA, Joan José**

SEMESTRE:

V

Huancayo-Perú

2021



RESUMEN

El presente proyecto consiste en implementar un sistema de reconocimiento de patrones numéricos utilizando un servidor web dada por Heroku a través de su plataforma como servicio para alojar aplicaciones. El funcionamiento del sistema se basa en el relleno de patrones en forma numérica por parte del usuario, reconocerlos, guardarlos en un arreglo y mostrar el resultado al reconocer el número ingresado.

El sistema tiene la capacidad de reconocer los números que son ingresados por el usuario gracias al entrenamiento de redes neuronales en el cual se utilizaron 3 patrones de cada número del 0 al 9 para un correcto entrenamiento de las redes neuronales.

Para un mejor entendimiento y comprensión del sistema, se le mostrará la codificación de la aplicación completa y la presentación del funcionamiento del programa en el servidor web utilizando diversos tipos de lenguajes con el fin de presentar un mejor entorno de interacción para el usuario, adicionalmente se presentará algunas limitaciones encontradas en el funcionamiento y las conclusiones al finalizar con todo el proyecto.



INTRODUCCIÓN

En esta nueva etapa del siglo XXI se está empezando a considerar a la inteligencia artificial como la nueva revolución, corazón de lo que algunos llaman la industria 4.0. Pero como hechos que respaldan esta idea observamos que la implementación de una amplia gama de tecnologías tales como: el reconocimiento de voz y la transcripción de voz a texto, nos muestra que gracias a esto la interacción con nuestros ordenadores y dispositivos cada vez se realiza de forma más comunicativa e interactiva.

Es importante el conocer acerca de estas tecnologías y apoyar dentro de su desarrollo para que cada vez sea mejor la realización de las tecnologías de la información. así romper las barreras que han estado presentes durante mucho tiempo entre la tecnología y las personas.

Dentro de la IA, tenemos al Deep Learning que observo el comportamiento del cerebro humano y se inspiró en intentar reproducir este comportamiento de forma informática. Para poder realizarlo necesito llevar elementos de la neurociencia al ordenador, es así como podemos encontrar a las redes neuronales.

Una red neuronal está compuesta por neuronas artificiales que se modelan de tal forma que imiten el comportamiento de una neurona cerebral, puedan procesar información, poder así tener conexiones entre ellas lo cual nos lleve a realizar una sinapsis y obtener resultados a partir de eso.

En el presente informe se muestra el uso de redes neuronales en el reconocimiento de números a través de datos de entrada los cuales se almacenan dentro de un TDA, con el fin de lograr cada vez un entrenamiento en las neuronas más exacto a los patrones que puedan tener la simbolización de los números del 0 al 9.



CONCEPTOS UTILIZADOS

FLASK:

Es un “micro” Framework escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web bajo el patrón MVC. Al instalarlo tenemos las herramientas necesarias para crear una web funcional, pero si se necesita en algún momento una nueva funcionalidad hay un conjunto muy grande de extensiones (Plugins) que se instalan junto a Flask y lo van dotando de funcionalidad.

El patrón MVC es una manera de trabajar que permite diferenciar y separar lo que es el modelo de datos, la vista (página HTML) y el controlador (donde se gestiona las peticiones de la app web). (Duran, 2018)



TensorFlow:

Es una biblioteca de software de código abierto para computación numérica, que utiliza flujo de datos. Es una gran plataforma para construir y entrenar redes neuronales, que permiten detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos.

La arquitectura flexible de TensorFlow le permite implementar el cálculo a una o más CPU o GPU en equipos de escritorio, servidores o dispositivos móviles en una sola API.

Fue desarrollada originalmente por investigadores e ingenieros que trabajan en el equipo de Google Brain Team, con el propósito de llevar a cabo el aprendizaje automático y la investigación de redes neuronales profundas. (Buhigas, 2018)



Módulo Keras:

Keras es un framework de alto nivel para el aprendizaje, escrito en Python y capaz de correr sobre los frameworks TensorFlow, CNTK o Theano. Fue desarrollado con el objeto de facilitar un proceso de experimentación rápida.
(Utrera, 2018)



Función de Activación:

Una función de activación es una función que transmite la información generada por la combinación lineal de los pesos y las entradas es decir son la manera de transmitir la información por las conexiones de salida. La información puede transmitirse sin modificaciones, función identidad o bien que no transmita la información. Como objetivo es que la red neuronal sea capaz de resolver problemas cada vez mas complejos, las funciones de activación generalmente harán que los modelos sean no lineales.
(Telefonica DATA UNIT, 2018)

Entre las funciones de activación usadas en el programa se encuentran:



- **Sigmoid – Sigmoidal:**

La función sigmoide transforma los valores introducidos a una escala (0,1), donde los valores altos tienden de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a 0.

Características de la función sigmoide:

- ✓ Satura y mata el gradiente
- ✓ Lenta convergencia
- ✓ No está centrada en el cero
- ✓ Esta acotada entre 0 y 1
- ✓ Buen rendimiento en la última capa

$$f(x) = \frac{1}{1 - e^{-x}}$$

Función Sigmoide

- **Tanh - Tangent Hyperbolic- Tangente Hyperbolica:**

La función tangente hiperbólica transforma los valores introducidos a una escala (-1,1), donde los valores altos tienden de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a -1.

Características de la función tangente hiperbólica:

- ✓ Muy similar a la sigmoidal
- ✓ Satura y mata el gradiente
- ✓ Lenta convergencia
- ✓ Centrada en 0
- ✓ Esta acotada entre -1 y 1
- ✓ Se utiliza para decidir entre una opción y la contraria
- ✓ Buen desempeño en redes recurrentes

$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$

Función tangente hiperbólica

- **ReLU – Rectified Lineal Unit**

La función transforma los valores introducidos anulando los valores negativos y dejando los positivos tal u como entran.

Características de la función ReLU:

- ✓ Activacion Sparse(Solo se activa si son positivos)



- ✓ No está acotada
- ✓ Se puede morir demasiadas neuronas
- ✓ Se comporta bien con imágenes
- ✓ Buen desempeño en redes convolucionales

$$f(x) = \max(0, x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

Función ReLU

- **Softmax - Rectified Lineal Unit:**

La función Softmax transforma las salidas a una representación en forma de probabilidades, de tal manera que el sumatoria de todas las probabilidades de las salidas de 1

Características de la función Softmax:

- ✓ Se utiliza cuando queremos tener una representación en forma de probabilidades
- ✓ Esta acotada entre 0 y 1
- ✓ Muy diferenciable
- ✓ Se utiliza para normalizar tipos multclases
- ✓ Buen rendimiento en las últimas capas. (Calvo, 2018)

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$

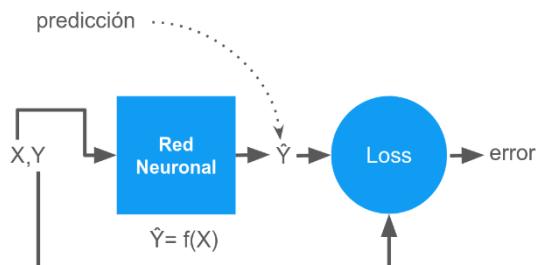
Función Softmax

Función de perdida:

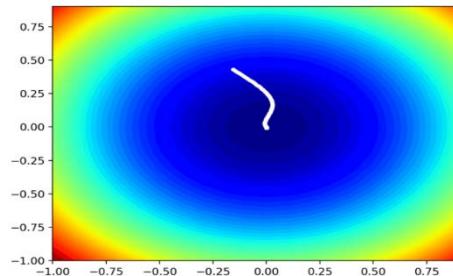
Una función de pérdida o Loss function es una función que evalúa la desviación entre las predicciones realizadas por la red neuronal y los valores reales de las observaciones utilizadas durante el aprendizaje. Cuando menor es el resultado de esta función, más eficiente es la red neuronal. Su minimización, es decir, reducir al mínimo la desviación entre el valor predicho y el valor real para una observación dada, se hace ajustando los distintos pesos de la red neuronal. (ENI, 2017)



Función de optimización Adán:



El algoritmo de Estimación de movimiento adaptativo, o Adán para abreviar, es una extensión del descenso de gradiente y un sucesor natural de técnicas como AdaGrad y RMSProp que adapta automáticamente una tasa de aprendizaje para cada variable de entrada para la función objetivo, suaviza aún más el proceso de entrada para la función objetivo y ayuda al proceso de búsqueda mediante el uso de una media móvil exponencialmente decreciente del gradiente para realizar actualizaciones a las variables. (BIGDATA, 2020)



Función de optimización SGD (descenso de gradiente estocástico):

Es una aproximación estocástica del gradiente descendente usado para minimizar una función objetivo que se escribe como una suma de funciones diferenciables. Este optimizador trata de encontrar mínimos o máximos por iteración.

Al igual que en la función del gradiente descendente, el gradiente indica la dirección en la que la función tiene la ratio de aumento más pronunciada, aunque no indica hasta donde se debe avanzar en esa dirección.

Ventajas del SGD:

- Si la función objetivo es la suma de costos individuales (errores) sobre un conjunto muy grande de datos. La muestra suele ser representativa y producir un valor muy cercano al de la población.
- Se reduce el número de cálculos en cada iteración.
- Cuando hay datos atípicos (*outliers*), las muestras pueden ser robustas a esas “pocas” grandes desviaciones (salvo en aquellas muestras que sean incluidos, que se esperan sean pocas).
- Si la función objetivo es (ruidosa, tiene muchos mínimos locales pequeños). El gradiente estocástico permite *suavizar* la función objetivo y reduce el riesgo de tener una convergencia temprana.



DESCRIPCIÓN DEL PROBLEMA:

1. Elaborar un programa en Python que utiliza una RED NEURONAL ARTIFICIAL para el reconocimiento de dígitos del 0 al 9 (ver más detalles en el anexo).

- a) El programa permitirá entrenar la red neuronal y permitirá reconocer un dígito suministrado al programa.
- b) La interfaz de usuario a utilizar para la entrada y salida de los datos lo define el grupo.
- c) Puede adaptar los programas que se implementó en clase para dar su solución.
- d) Si realiza un programa nuevo (desde cero) debe utilizar la librería Keras de Python para implementar las redes neuronales.
- e) Puede utilizar otras librerías que el grupo crea conveniente para la interfaz de usuario, lectura de datos, etc.

ESTRATEGIAS DE SOLUCIÓN:

Tenemos la opción de entregarle los datos de ingreso a la red neuronal de forma bidimensional en un arreglo de arreglo y también la forma de entregarle los datos en un arreglo unidimensional. En nuestro caso utilizamos el arreglo dimensional para ingresar los patrones de 35 números entre 0 y 1.

- **Patrones de ingreso de datos:**

```
number_patterns_secondary.txt
```

0,1,1,1,0,1,0,0,0,1,1,0,0,1,1,1,0,1,0,1,1,0,0,1,1,0,0,0,1,0,0,1,1,1,0
1,1,1,1,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,1,1,1,1

0,0,1,0,0,0,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,1,0
0,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0

0,1,1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1,0
1,1,1,1,1,1,0,0,0,1,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,0,1,1,0,0,1,1,1,1,1

1,1,1,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,0,0,0,1,1,1,0
1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1
0,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,1,1

0,0,0,1,0,0,0,1,1,0,0,0,1,0,0,0,1,0,0,0,1,0,1,1,1,1,0,0,0,0,1,0,0,0,0,1,0
1,0,0,0,1,1,0,0,0,1,1,0,0,0,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0
0,0,0,1,0,0,0,0,1,1,0,0,0,1,0,0,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0

1,1,1,1,1,1,0,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,1,1,1,1,0
1,1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,1,1,0,0,0,0,1,0,0,0,1,1,1,1,1,0

0,0,1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,1,1,1,0,1,0,0,0,0,1,1,0,0,0,1,1,1,0
0,0,1,1,1,0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,1,1,0,0,0,0,1,1,1,1,1,1
1,1,1,1,1,1,0,0,0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,1,1,0,0,0,1,1,1,1,1

1,1,1,1,1,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0
1,1,1,1,1,0,0,0,0,0,1,0,0,0,0,1,1,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,1,0,0,0

0,1,1,1,1,0,0,0,0,1,1,0,0,0,1,0,1,1,1,1,0,1,0,0,0,0,1,1,0,0,0,1,0,1,1,1,0
1,1,1,1,1,0,0,0,0,1,1,0,0,0,1,1,1,1,1,1,0,1,0,0,0,0,1,1,0,0,0,1,1,1,1,1,0

0,1,1,1,1,0,0,0,0,1,1,0,0,0,1,0,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,1,0,0,1,1,0,0
1,1,1,1,1,0,0,0,0,1,1,0,0,0,1,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,0,1,1,1,1,1,0
1,1,1,1,1,0,0,0,0,1,1,0,0,0,1,1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,1

- El patrón de ingreso de datos esta dado por 35 números entre 0 y 1.



- **Patrones de salida de datos:**

```
number_target_primary.txt
1,0,0,0,0,0,0,0,0,0
1,0,0,0,0,0,0,0,0,0

0,1,0,0,0,0,0,0,0,0
0,1,0,0,0,0,0,0,0,0

0,0,1,0,0,0,0,0,0,0
0,0,1,0,0,0,0,0,0,0

0,0,0,1,0,0,0,0,0,0
0,0,0,1,0,0,0,0,0,0

0,0,0,0,1,0,0,0,0,0
0,0,0,0,1,0,0,0,0,0

0,0,0,0,0,1,0,0,0,0
0,0,0,0,0,1,0,0,0,0

0,0,0,0,0,0,1,0,0,0
0,0,0,0,0,0,1,0,0,0

0,0,0,0,0,0,0,1,0,0
0,0,0,0,0,0,0,1,0,0

0,0,0,0,0,0,0,0,1,0
0,0,0,0,0,0,0,0,1,0
```

- El patrón de salida este dado por números de 0 y un solo 1, que este último representa en posición al dígito que debe predecir.

EXPLICACIÓN GENERAL:

Primero compilamos la red definiendo una función de pérdida y un optimizador: en nuestro caso seleccionamos “categorical_crossentropy”, porque tenemos múltiples categorías (como en los números 0-9). La función *number_recognition_model* nos crea un modelo de *Sequential* para la predicción de números expresados en patrones de 35 números entre 0 y 1 esta función genera tres capas la primera con 32 neuronas con función de activación "relu" otra con el doble de la anterior con 64 y también función de activación "relu" y la capa de salida con 10 neuronas con función de activación "softmax".

TESTING:

Utilizamos distintas variantes para mejorar y obtener la mejor red neuronal para la aplicación:

Optimizadores:

- Adam
- SGD

Funciones de activación:

- Relu
- Sigmoid
- Tanh
- Softmax



Función Dropout:

- Esta es una técnica de regularización para reducir el sobreajuste en redes neuronales artificiales.

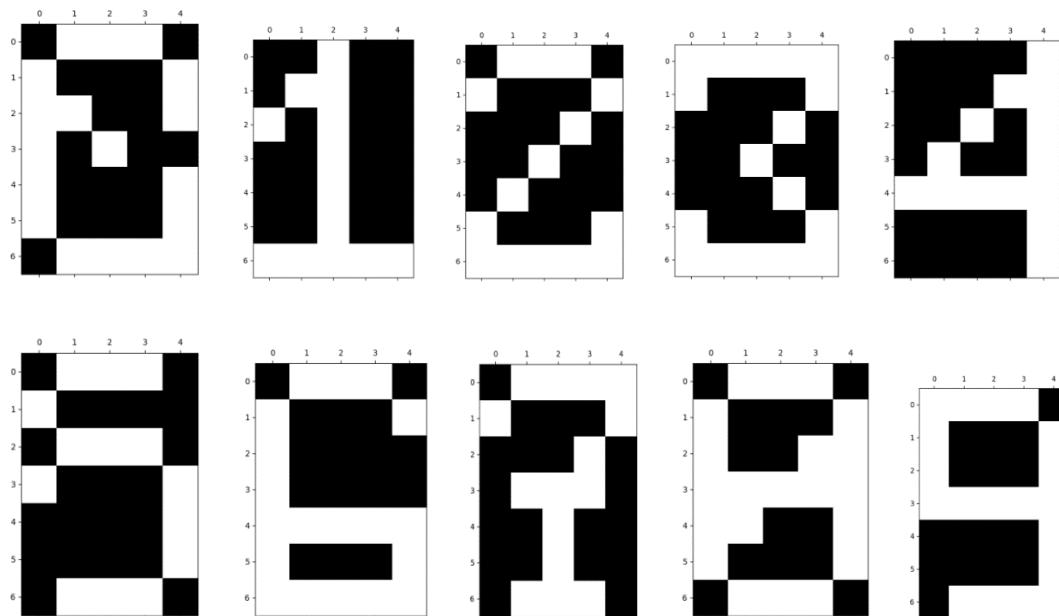
Función de pérdida:

- Category crossentropy

Datos diferentes a los patrones de entrenamiento:

```
testing_data_primary.txt
testing_data_primary.txt
1 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1
2 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1
3 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1
4 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1
5 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1
6 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
7 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
8 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
9 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1
10 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1
11 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1
12 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1
13 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1
14 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1
15 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1
16 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1
17 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1
18 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1
19 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1
20 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1
```

✓ La representación de estos patrones en un arreglo bidimensional es:





SIN DROPOUT:

- En estos casos entre 300 y 700 epochs obtuvimos buenos resultados.
 - Utilizando “relu” en las capas ocultas y “softmax” en capa de salida”.

1. Optimizador “adam” con 500 epochs.

```
7 != 10
Expected :10
Actual   :7
<Click to see difference>
Traceback (most recent call last):
  File "/home/master/Programming/maching-learning/src/recognition_number/patterns.py", line 65, in test_prediction
    self.assertEqual(10, count)
AssertionError: 10 != 7
```

- Pasa 7 / 10 de los casos de prueba con una precisión aproximada del 70 %.

2. Optimizador “sgd” con 500 epochs:

```
78 def number_recognition_model():
79     # dropout = Dropout(0.29)
80     layer_one = Dense(32, activation='relu')
81     layer_two = Dense(64, activation='relu')
82     layer_three = Dense(10, activation='softmax')
83
84     number_model = Sequential()
85     # number_model.add(dropout)
86     number_model.add(layer_one)
87     number_model.add(layer_two)
88     number_model.add(layer_three)
89     number_model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
90
91     return number_model
```



UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

FACULTAD DE INGENIERÍA DE SISTEMAS



```
Run: Unitests in patterns.py
  Test Results
    patterns
      TestCasePrediction
        test_prediction
          Tests failed: 1 of 1 test - 5 ms
          0.09356312 round fixed => 0.09 percentual => 9.36%
          [3.8491347e-04 1.0630898e-02 6.2780701e-02 7.5553678e-02 2.0524107e-03
           1.1849184e-02 2.9217063e-03 8.0929208e-01 2.7297509e-03 2.1894572e-02]
          0.8897621 round fixed => 0.81 percentual => 88.92%
          [2.7989319e-01 3.2543840e-03 1.6868563e-02 1.5165419e-03 2.6174728e-02
           7.7721616e-03 3.3518281e-02 3.4468356e-04 5.6208330e-01 6.8574227e-02]
          0.5620833 round fixed => 0.56 percentual => 56.21%
          [0.01306745 0.00107601 0.01371951 0.02978087 0.04470887 0.03711582
           0.02359771 0.00057816 0.2899915 0.5463641]
          0.5463641 round fixed => 0.55 percentual => 54.64%
          test accuracy => 70.0%
          Ran 1 test in 0.005s
          FAILED (failures=1)
          7 != 10
          Expected :10
          Actual   :7
          <Click to see difference>
          Traceback (most recent call last):
            File "/home/master/Programming/maching-learning/src/recognition_number/patterns.py", line 65, in test_prediction
              self.assertEqual(10, count)
            AssertionError: 10 != 7
```

- Pasa 7 / 10 de los casos de prueba con una precisión aproximada del 70 %.

CON “SIGMOID” EN LAS CAPAS OCULTAS, “SOFTMAX” EN CAPA DE SALIDA”:

1. Optimizador “adam” con 500 epochs:

```
Run: Unitests in patterns.py
  Test Results
    patterns
      TestCasePrediction
        test_prediction
          Tests failed: 1 of 1 test - 5 ms
          8.8883225e-01 6.5281312e-03 9.1095865e-03 2.2356706e-03 5.3822988e-03]
          0.88983225 round fixed => 0.88 percentual => 88.08%
          [4.3492550e-01 1.2543054e-01 5.0362654e-02 5.2978756e-04 3.1104948e-02
           1.1087163e-02 2.8826943e-01 2.5388762e-03 5.6881305e-02 2.30862807e-04]
          0.28826945 round fixed => 0.29 percentual => 28.83%
          [1.5522637e-02 2.79222314e-02 7.2219744e-02 3.6186416e-02 3.1426044e-05
           1.8805906e-02 6.9278572e-04 7.5976729e-01 8.9470996e-03 5.7593258e-02]
          0.7597673 round fixed => 0.76 percentual => 75.98%
          [4.1719633e-01 4.1835825e-03 1.6843317e-02 3.8495667e-04 4.7836096e-02
           2.4328368e-02 1.8170470e-02 1.9417817e-03 4.3743542e-01 3.1677812e-02]
          0.43743542 round fixed => 0.44 percentual => 43.74%
          [2.3629155e-02 2.69805324e-04 7.5029498e-03 6.2018256e-03 1.9575872e-02
           6.0636863e-02 1.6873218e-03 1.1577129e-02 1.4659881e-01 7.2232896e-01]
          0.77232896 round fixed => 0.72 percentual => 72.23%
          test accuracy => 80.0%
          8 != 10
          Expected :10
          Actual   :8
          <Click to see difference>
          Traceback (most recent call last):
            File "/home/master/Programming/maching-learning/src/recognition_number/patterns.py", line 65, in test_prediction
              self.assertEqual(10, count)
            AssertionError: 10 != 8
```

- Pasa 8 / 10 de los casos de prueba con una precisión aproximada del 80 %.



2. Optimizador “sgd” con 500 epochs:

```
78     def number_recognition_model():
79         # dropout = Dropout(0.20)
80         layer_one = Dense(32, activation='sigmoid')
81         layer_two = Dense(64, activation='sigmoid')
82         layer_three = Dense(10, activation='softmax')
83
84         number_model = Sequential()
85         # number_model.add(dropout)
86         number_model.add(layer_one)
87         number_model.add(layer_two)
88         number_model.add(layer_three)
89         number_model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
90
91     return number_model
```

Tests failed: 1 of 1 test ~ 4 ms

- 0.09863713 0.09431104 0.09816412 0.10064778]

0.098637134 round fixed => 0.1 porcentual => 9.86%

[0.0978066 0.10347792 0.09746561 0.09956317 0.0971137 0.0997666 0.10059942 0.1013846 0.1042517 0.09857867]

0.101384595 round fixed => 0.1 porcentual => 10.14%

[0.10296363 0.09679222 0.10398974 0.09997176 0.0963282 0.09851769 0.10346574 0.09703413 0.09859626 0.10294264]

0.09839626 round fixed => 0.1 porcentual => 9.84%

[0.10232133 0.0988143 0.09908936 0.09990129 0.09862304 0.09993113 0.10148368 0.09895573 0.10140162 0.09947857]

0.09947857 round fixed => 0.1 porcentual => 9.95%

test accuracy => 0.0%

Ran 1 test in 0.005s

FAILED (failures=1)

0 != 10

Expected :10
Actual :0
[Click to see difference](#)

Traceback (most recent call last):
File "/home/master/Programming/machine-learning/src/recognition_number/patterns.py", line 65, in test_prediction
self.assertEqual(10, count)
AssertionError: 10 != 0

- Pasa 0 / 10 de los casos de prueba con una precisión aproximada del 0%.

CON “TANH” EN LAS CAPAS OCULTAS, “SOFTMAX” EN CAPA DE SALIDA:

1. Optimizador “adam” con 500 epochs:

```
78     def number_recognition_model():
79         # dropout = Dropout(0.20)
80         layer_one = Dense(32, activation='tanh')
81         layer_two = Dense(64, activation='tanh')
82         layer_three = Dense(10, activation='softmax')
83
84         number_model = Sequential()
85         # number_model.add(dropout)
86         number_model.add(layer_one)
87         number_model.add(layer_two)
88         number_model.add(layer_three)
89         number_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
90
91     return number_model
```



UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

FACULTAD DE INGENIERÍA DE SISTEMAS



The screenshot shows the PyCharm interface with the 'Test Results' tool window open. It displays the output of a test named 'test_prediction'. The output includes several numerical comparisons between expected and actual values, some of which are highlighted in red. A failure message is shown at the bottom:

```
Expected :10
Actual   :8
<Click to see difference>
Traceback (most recent call last):
  File "/home/master/Programming/maching-learning/src/recognition_number/patterns.py", line 65, in test_prediction
    self.assertEqual(10, count)
AssertionError: 10 != 8
```

- Pasa 8 / 10 de los casos de prueba con una precisión aproximada del 80%.

2. Optimizador "sgd" con 500 epochs:

The screenshot shows the PyCharm interface with the 'Test Results' tool window open. It displays the output of a test named 'test_prediction'. The output includes several numerical comparisons between expected and actual values, some of which are highlighted in red. A failure message is shown at the bottom:

```
Expected :10
Actual   :7
<Click to see difference>
Traceback (most recent call last):
  File "/home/master/Programming/maching-learning/src/recognition_number/patterns.py", line 65, in test_prediction
    self.assertEqual(10, count)
AssertionError: 10 != 7
```

- Pasa 7 / 10 de los casos de prueba con una precisión aproximada del 70%.



CON DROPOUT AL 20%:

- Ya que trabajaremos con un Dropout debemos entrenar la red con más epochs en este caso notamos que llegaban a buenos entrenamientos entre 3000 y 6000 epochs.

CON “RELU” EN LAS CAPAS OCULTAS, “SOFTMAX” EN CAPA DE SALIDA”:

1. Optimizador “adam” con 4000 epochs:

```
78     def number_recognition_model():
79         dropout = Dropout(0.20)
80         layer_one = Dense(32, activation='relu')
81         layer_two = Dense(64, activation='relu')
82         layer_three = Dense(10, activation='softmax')
83
84         number_model = Sequential()
85         number_model.add(dropout)
86         number_model.add(layer_one)
87         number_model.add(layer_two)
88         number_model.add(layer_three)
89         number_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
90
91     return number_model
```

The screenshot shows the PyCharm IDE interface with the code editor and a terminal window. The terminal displays the following test results:

```
Tests failed: 1 of 1 test - 5 ms
  9.9987900e-01 3.8819966e-08 4.8383104e-08 1.5344602e-10 2.3355724e-06
  0.999879 round fixed => 1.0 porcentual => 99.99%
[3.7283504e-01 3.4972087e-09 4.4172998e-06 3.9751849e-06 1.4133774e-06
 5.8530380e-07 6.2646878e-01 1.7292603e-08 6.8582812e-04 3.0857593e-09]
  0.6264688 round fixed => 0.63 porcentual => 62.65%
[2.3035449e-18 7.8848569e-04 1.7137198e-03 1.3887877e-05 4.4692858e-09
 4.7691210e-06 3.5088508e-06 9.9747854e-01 1.2740513e-07 4.8832508e-06]
  0.99747854 round fixed => 1.0 porcentual => 99.75%
[8.0549043e-01 2.7320890e-08 1.8763814e-05 6.9518768e-07 1.0735155e-06
 2.7136124e-07 4.4197589e-05 1.9631927e-09 1.8994832e-01 1.2070919e-04]
  0.18994832 round fixed => 0.19 porcentual => 18.99%
[7.8794440e-06 3.5147235e-10 4.8719735e-06 2.5191324e-05 4.0233566e-08
 4.7330999e-05 1.0834514e-06 5.8899735e-10 9.6617234e-02 9.0369723e-01]
  0.9838972 round fixed => 0.9 porcentual => 98.39%
test accuracy => 80.0%
  8 != 10
    Expected :10
    Actual   :8
    <Click to see difference>
Traceback (most recent call last):
  File "/home/master/Programming/maching-learning/src/recognition_number/patterns.py", line 65, in test_prediction
    self.assertEqual(10, count)
AssertionError: 10 != 8
```

- Pasa 8 / 10 de los casos de prueba con una precisión aproximada del 80%.

2. Optimizador “sgd” con 4000 epochs:

```
def number_recognition_model():
    dropout = Dropout(0.20)
    layer_one = Dense(32, activation='relu')
    layer_two = Dense(64, activation='relu')
    layer_three = Dense(10, activation='softmax')

    number_model = Sequential()
    number_model.add(dropout)
    number_model.add(layer_one)
    number_model.add(layer_two)
    number_model.add(layer_three)
    number_model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

    return number_model
```



UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

FACULTAD DE INGENIERÍA DE SISTEMAS



```
Tests failed: 1 of 1 test - 6ms
0.72355217 round fixed => 0.72 porcentual => 72.36%
[1.57189944e-05 9.76427307e-03 5.49166948e-02 6.46955445e-02
 1.01011734e-04 2.90294131e-03 3.35067837e-03 8.57224882e-01
 3.26428205e-03 3.82481375e-03]

0.8572249 round fixed => 0.86 porcentual => 85.72%
[2.6010761e-01 5.2355326e-07 2.2859068e-04 6.0811308e-06 4.1410804e-05
 1.2989518e-05 1.5994053e-02 1.641818e-07 7.2161764e-01 1.9908897e-03]

0.72161764 round fixed => 0.72 porcentual => 72.16%
[3.2055835e-04 7.6363861e-08 2.4874169e-04 2.1681543e-04 1.5399228e-04
 1.5368644e-03 1.7940416e-04 3.9530383e-07 1.8320936e-01 8.1424361e-01]

0.8142436 round fixed => 0.81 porcentual => 81.42%
test accuracy => 98.0%

Ran 1 test in 0.007s
FAILED (failures=1)

9 != 10
Expected :10
Actual   :9
<Click to see difference>

Traceback (most recent call last):
  File "/home/master/Programming/maching-learning/src/recognition_number/patterns.py", line 65, in test_prediction
    self.assertEqual(10, count)
AssertionError: 10 != 9
```

- Pasa 9 / 10 de los casos de prueba con una precisión aproximada del 90%.

CON “SIGMOID” EN LAS CAPAS OCULTAS, “SOFTMAX” EN CAPA DE SALIDA”:

1.Optimizador “adam” con 4000 epochs:

```
def number_recognition_model():
    dropout = Dropout(0.20)
    layer_one = Dense(32, activation='sigmoid')
    layer_two = Dense(64, activation='sigmoid')
    layer_three = Dense(10, activation='softmax')

    number_model = Sequential()
    number_model.add(dropout)
    number_model.add(layer_one)
    number_model.add(layer_two)
    number_model.add(layer_three)
    number_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

    return number_model

Tests failed: 1 of 1 test - 5ms
9.9886863e-01 9.0119784e-06 2.2635589e-05 2.3102334e-06 3.8148425e-04
9.9886863 round fixed => 1.0 porcentual => 99.81%
[0.2549480e-02 5.3208944e-04 2.2137181e-05 2.5873682e-05 1.8044362e-04
 3.9892137e-05 9.0598565e-01 5.4053686e-09 6.6438783e-04 8.5625651e-08]

0.98598565 round fixed => 0.91 porcentual => 98.6%
[1.4350566e-07 6.2092408e-03 2.3059766e-03 4.7991201e-03 2.8451570e-05
 1.7673759e-04 9.9671170e-06 9.8663165e-01 9.9573833e-07 4.3783351e-04]

0.98603165 round fixed => 0.99 porcentual => 98.6%
[7.3749959e-01 1.8768656e-04 1.8337631e-04 2.1635115e-03 3.7369115e-04
 2.8984604e-04 1.7334655e-02 3.1594723e-08 2.4156614e-01 3.9948886e-04]

0.24156614 round fixed => 0.24 porcentual => 24.16%
[1.1248013e-04 1.1048727e-05 4.3988195e-05 5.9616350e-04 5.1414860e-05
 6.5311766e-04 4.2512956e-06 1.1464292e-06 7.5417787e-02 9.2310876e-01]

0.92310876 round fixed => 0.92 porcentual => 92.31%
test accuracy => 80.0%

8 != 10
Expected :10
Actual   :8
<Click to see difference>

Traceback (most recent call last):
  File "/home/master/Programming/maching-learning/src/recognition_number/patterns.py", line 65, in test_prediction
    self.assertEqual(10, count)
AssertionError: 10 != 8
```

- Pasa 8 / 10 de los casos de prueba con una precisión aproximada del 80%.



2.Optimizador “sgd” con 4000 epochs:

```
78     def number_recognition_model():
79         dropout = Dropout(0.20)
80         layer_one = Dense(32, activation='sigmoid')
81         layer_two = Dense(64, activation='sigmoid')
82         layer_three = Dense(10, activation='softmax')
83
84         number_model = Sequential()
85         number_model.add(dropout)
86         number_model.add(layer_one)
87         number_model.add(layer_two)
88         number_model.add(layer_three)
89         number_model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
90
91     return number_model
```

Tests failed: 1 of 1. test - 6 ms

0.09876633 0.06438965 0.08296745 0.09980484

0.135784 round fixed => 0.14 porcentual => 15.57%

[0.15218122 0.01937944 0.14288895 0.0691167 0.07252628 0.18425536 0.1807955 0.01285001 0.1581021 0.0879923]

0.1807955 round fixed => 0.18 porcentual => 18.08%

[0.03683442 0.15855444 0.07974375 0.14471422 0.11440492 0.86848725 0.03971802 0.2644964 0.03212671 0.06091962]

0.2644964 round fixed => 0.26 porcentual => 26.45%

[0.15456902 0.81057188 0.11865974 0.07062441 0.07697795 0.18895731 0.15911736 0.01497898 0.16793485 0.11762831]

0.16793485 round fixed => 0.17 porcentual => 16.79%

[0.13921873 0.81012762 0.1159697 0.0879681 0.08539517 0.12636203 0.11553569 0.02459487 0.13966922 0.15515894]

0.15515894 round fixed => 0.16 porcentual => 15.52%

test accuracy => 0.0%

0 != 10

Expected :10
Actual :0
<Click to see differences>

Traceback (most recent call last):
File "/home/master/Programming/maching-learning/src/recognition_number/patterns.py", line 65, in test_prediction
self.assertEqual(10, count)
AssertionError: 10 != 0

- Pasa 0 / 10 de los casos de prueba con una precisión aproximada del 0%.

CON “TANH” EN LAS CAPAS OCULTAS, “SOFTMAX” EN LA CAPA DE SALIDA:

1.Optimizador “adam” con 4000 epochs:

```
78     def number_recognition_model():
79         dropout = Dropout(0.20)
80         layer_one = Dense(32, activation='tanh')
81         layer_two = Dense(64, activation='tanh')
82         layer_three = Dense(10, activation='softmax')
83
84         number_model = Sequential()
85         number_model.add(dropout)
86         number_model.add(layer_one)
87         number_model.add(layer_two)
88         number_model.add(layer_three)
89         number_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
90
91     return number_model
```



UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

FACULTAD DE INGENIERÍA DE SISTEMAS



```
Tests failed: 1 of 1 test - 5ms
9.9982268e-01 5.0686668e-07 1.9013835e-06 9.5810975e-08 2.2661065e-05]
0.9998227 round fixed => 1.0 porcentual => 99.98%
[6.7755058e-02 2.6340137e-04 2.8073202e-06 1.8625076e-06 4.3756492e-05
1.2900816e-06 9.3179071e-01 3.1256261e-07 1.4158509e-04 3.0237597e-09]
0.9317907 round fixed => 0.93 porcentual => 93.18%
[1.2206814e-01 1.9778986e-04 1.4755783e-04 2.3465884e-05 8.6599894e-08
1.5955469e-05 6.0882385e-06 9.9944097e-01 1.6294352e-07 1.6799085e-04]
0.99944097 round fixed => 1.0 porcentual => 99.94%
[6.9666386e-01 1.8119825e-05 2.5468651e-05 1.8688701e-04 4.8957610e-04
4.8484660e-07 4.1809976e-03 1.7729247e-06 2.9846688e-01 5.5758675e-05]
0.2984668 round fixed => 0.3 porcentual => 29.85%
[8.3392397e-06 1.2121343e-07 3.1503343e-05 8.5312982e-05 2.7713620e-06
6.6117558e-05 1.4095865e-06 4.3717080e-07 3.1436479e-01 6.8543917e-01]
0.68543917 round fixed => 0.69 porcentual => 68.54%
test accuracy => 70.0%
7 != 10
Expected :10
Actual :7
<Click to see difference>
Traceback (most recent call last):
  File "/home/master/Programming/maching-learning/src/recognition_number/patterns.py", line 65, in test_prediction
    self.assertEqual(10, count)
AssertionError: 10 != 7
```

- Pasa 7 / 10 de los casos de prueba con una precisión aproximada del 70%.

2.Optimizador "sgd" con 4000 epochs:

```
def number_recognition_model():
    dropout = Dropout(0.20)
    layer_one = Dense(32, activation='tanh')
    layer_two = Dense(64, activation='tanh')
    layer_three = Dense(10, activation='softmax')

    number_model = Sequential()
    number_model.add(dropout)
    number_model.add(layer_one)
    number_model.add(layer_two)
    number_model.add(layer_three)
    number_model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

    return number_model
Tests failed: 1 of 1 test - 5ms
9.5926815 round fixed => 0.96 porcentual => 95.93%
[5.3573889e-01 3.2880637e-04 1.8642455e-03 1.1858718e-03 2.5079935e-04
4.9146666e-04 4.1078210e-01 2.0347998e-06 4.9308117e-02 4.9644328e-05]

0.4107821 round fixed => 0.41 porcentual => 41.08%
[2.75563343e-05 4.80083935e-02 5.77959493e-02 5.29556395e-05
2.77672836e-04 4.94623370e-03 1.17376236e-04 8.55621815e-01
8.64275848e-04 2.70451680e-02]

0.8556218 round fixed => 0.86 porcentual => 85.56%
[3.5274076e-01 3.0454091e-04 3.2882565e-03 1.0335350e-03 3.7685467e-03
5.1614665e-04 2.5060579e-02 4.6001496e-05 6.0446733e-01 8.7743215e-03]

0.60446733 round fixed => 0.6 porcentual => 60.45%
[1.7072052e-03 1.1284593e-04 1.3984489e-03 1.2130311e-03 3.0894955e-03
7.9317987e-03 3.8379565e-04 3.7600141e-04 1.6970460e-01 8.1400274e-01]

0.81408274 round fixed => 0.81 porcentual => 81.41%
test accuracy => 90.0%

9 != 10
Expected :10
Actual :9
<Click to see difference>
Traceback (most recent call last):
  File "/home/master/Programming/maching-learning/src/recognition_number/patterns.py", line 65, in test_prediction
    self.assertEqual(10, count)
AssertionError: 10 != 9
```



CASOS PECULIARES:

1. Usando solo la función de activación sigmoid sin Dropout y 500 epochs:

```
78 def number_recognition_model():
79     # dropout = Dropout(0.20)
80     layer_one = Dense(32, activation='sigmoid')
81     layer_two = Dense(64, activation='sigmoid')
82     layer_three = Dense(10, activation='sigmoid')
83
84     number_model = Sequential()
85     # number_model.add(dropout)
86     number_model.add(layer_one)
87     number_model.add(layer_two)
88     number_model.add(layer_three)
89     number_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
90
91     return number_model
```

Tests passed: 1 of 1 test - 5 ms

Test Results

patterns

TestCasePrediction

test_prediction

[0.2249316 0.5782993 0.99452746 0.9868741 0.03341097 0.9597095
0.13758439 0.6983507 0.6105902 0.20115593]

0.9868741 round fixed => 0.99 porcentual => 98.69%

[0.02290654 0.17558765 0.00140217 0.7049333 0.9982418 0.42563954
0.8867878 0.9236498 0.2559735 0.4259832]

0.9982418 round fixed => 1.0 porcentual => 99.82%

[0.8884896 0.770494 0.6209379 0.1728509 0.1737954 0.9929281
0.24464047 0.25257128 0.23374942 0.57936084]

0.9929281 round fixed => 0.99 porcentual => 99.29%

[0.9960022 0.1202338 0.39355442 0.05843896 0.45096198 0.87481403
0.95382696 0.01353601 0.98461753 0.4118656]

0.95382696 round fixed => 0.95 porcentual => 95.38%

[0.01500329 0.96041334 0.9630138 0.8636384 0.25805604 0.8389622
0.00788289 0.98464745 0.05531299 0.8289679]

0.98464745 round fixed => 0.98 porcentual => 98.46%

[0.9844637 0.04009798 0.13991237 0.07286677 0.77616346 0.7430284
0.92176056 0.03837854 0.9916101 0.80886436]

0.9916101 round fixed => 0.99 porcentual => 99.16%

[0.8624321 0.3162129 0.75839514 0.09537983 0.5021598 0.80591196
0.08502948 0.26937413 0.9111542 0.9892896]

0.9892896 round fixed => 0.99 porcentual => 98.93%

test accuracy => 100.0%

- Este resultado es peculiar debido a que distribuye un valor aproximado de 1 a los que se parecen a dicho patrón.

2. Usando solo función de activación sigmoid con Dropout y 5000 epochs:

```
78 def number_recognition_model():
79     dropout = Dropout(0.20)
80     layer_one = Dense(32, activation='sigmoid')
81     layer_two = Dense(64, activation='sigmoid')
82     layer_three = Dense(10, activation='sigmoid')
83
84     number_model = Sequential()
85     number_model.add(dropout)
86     number_model.add(layer_one)
87     number_model.add(layer_two)
88     number_model.add(layer_three)
89     number_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
90
91     return number_model
```



```
Tests passed: 1 of 1 test - 4 ms
[0.2793397  0.00804645  0.56815356  0.9992699  0.00198159  0.7199404
 0.68340707  0.06945264  0.23656081  0.00429279]
0.9992699 round fixed => 1.0 porcentual => 99.93%
[0.09429577  0.0306437   0.02007416  0.9084795  0.99997187  0.00185251
 0.8819635   0.8123865   0.00164822  0.02154982]
0.99997187 round fixed => 1.0 porcentual => 100.0%
[7.21651852e-01 7.33391464e-01 1.67757273e-04 8.47423911e-01
 4.70199883e-02 9.99924541e-01 1.24471545e-01 7.09152222e-03
 1.88419223e-03 5.76998234e-01]
0.99992454 round fixed => 1.0 porcentual => 99.99%
[9.9182057e-01 3.9859596e-01 7.3983043e-02 4.9161315e-02 6.8113190e-01
 9.3980402e-02 9.9951410e-01 1.0030649e-04 7.3171157e-01 4.7945976e-04]
0.9995141 round fixed => 1.0 porcentual => 99.95%
[0.00106645  0.9132817   0.8898766   0.5069959   0.04180557  0.15232956
 0.15042716  0.9987949   0.00143865  0.4367178  ]
0.9987949 round fixed => 1.0 porcentual => 99.88%
[9.9353576e-01 4.1188002e-03 2.3124993e-01 2.2165385e-01 5.5963618e-01
 1.5735477e-02 9.5381731e-01 2.3952127e-04 9.9555647e-01 2.4179956e-01]
0.9955565 round fixed => 1.0 porcentual => 99.56%
[1.8989250e-01 1.4776289e-03 5.9391409e-02 4.3069673e-01 1.4972183e-01
 6.2645286e-01 9.3098342e-02 9.9304318e-04 9.9537051e-01 9.9936199e-01]
0.999362 round fixed => 1.0 porcentual => 99.94%
test accuracy => 100.0%
```

- Resultado similar que distribuye a los números posibles un valor aproximado de 1.

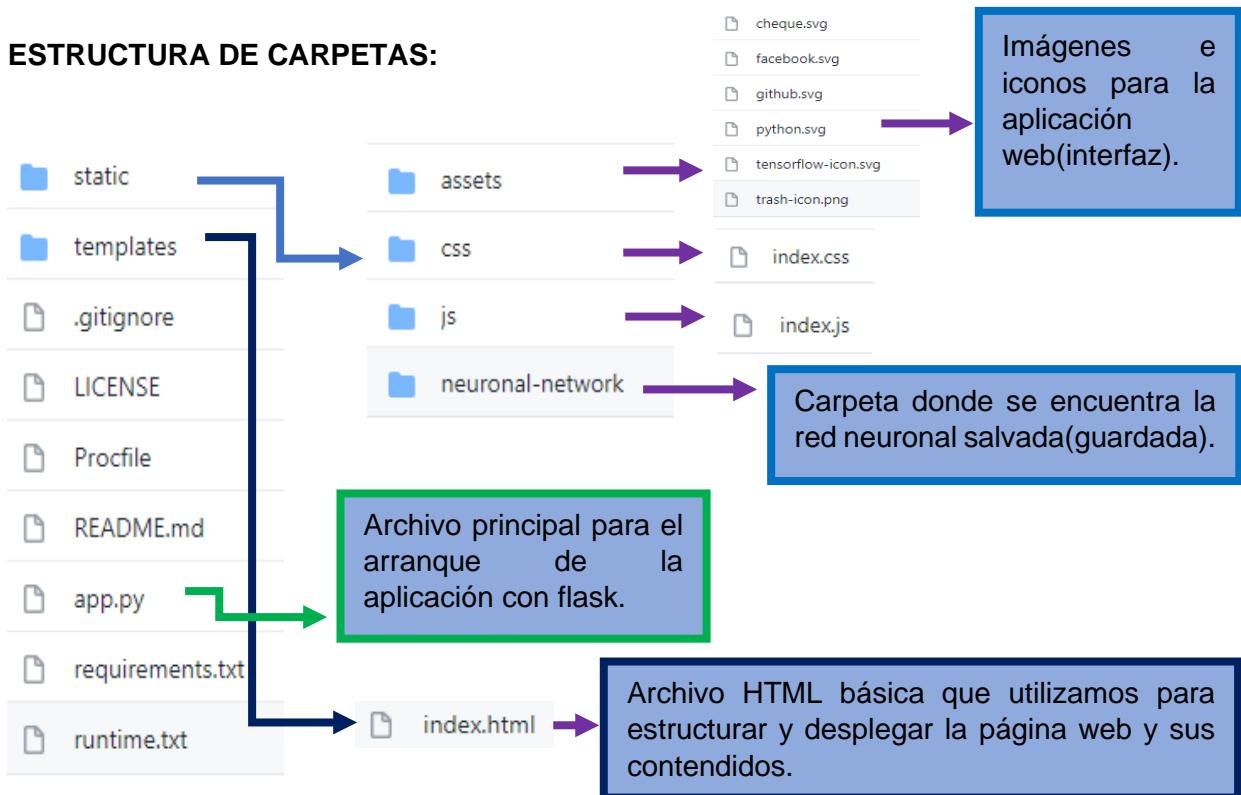
Finalizando las distintas pruebas con la red neuronal:

- Para la aplicación web usamos todas las redes neuronales que llegaron a los 9 / 10 casos de prueba siendo con las funciones de activación tanh y relu en capas ocultas, y de salida softmax las que obtuvieron los mejores resultados, para una predicción muy buena solo teniendo entre dos a tres patrones de entrenamiento por cada número.

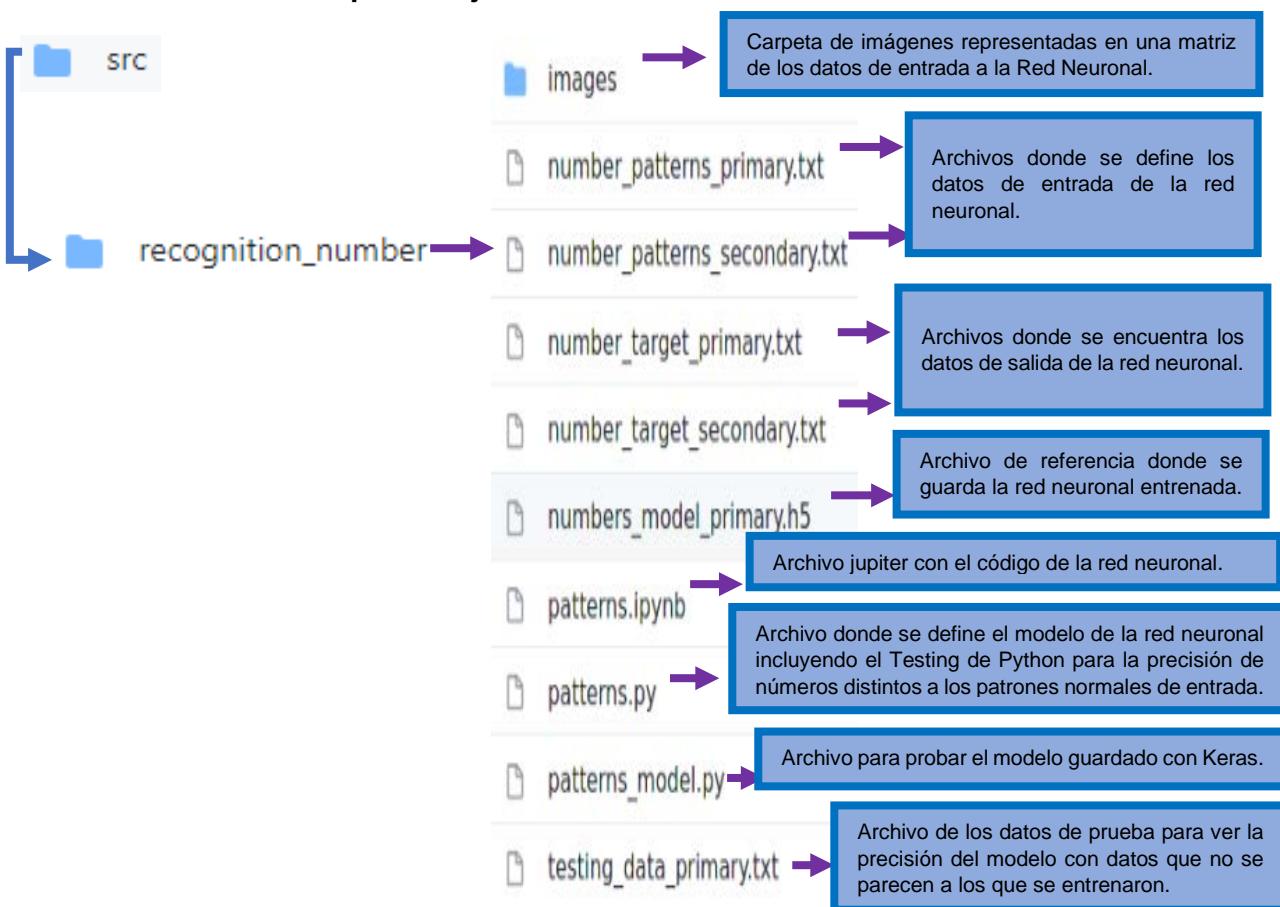


CÓDIGO FUENTE DEL PROGRAMA EN PYTHON:

ESTRUCTURA DE CARPETAS:



Dentro del entorno de aprendizaje:





INFORMACIÓN DEL LENGUAJE UTILIZADO PARA EL DESARROLLO DE LA RED NEURONAL:

| LENGUAJE DE PROGRAMACIÓN | ÍCONO | VERSIÓN | EXPLICACIÓN |
|--------------------------|-------|---------|--|
| Python | | 3.9.5 | Utilizado tanto para el desarrollo como despliegue de la aplicación. |

CODIFICACIÓN EN PYTHON:

| SUBCARPETA | ARCHIVO |
|--------------------|-------------|
| recognition_number | patterns.py |

`import os`

Nos permite ocultar mensajes innecesarios provocados por Tensorflow

Nos permite realizar los test de nuestra aplicación.

`import unittest`

`import time`

Nos permite manejar el tiempo de entrenamiento

Nos permite realizar gráficos en 2D. Asignamos la variable pl.

`import pylab as pl`

`import numpy as np`

Nos permite procesar las matrices. Asignamos la variable np.

Nos permite graficar el muestreo de los datos. Asignamos la variable plt.

`import matplotlib.pyplot as plt`



Del módulo de models en Keras importamos la clase Sequential apropiado para una pila simple de capas donde cada capa tiene un tensor de entrada y de salida.

```
from keras.models import Sequential
```

```
from keras.layers import Dense, Dropout
```

Del módulo de layers en Keras importamos la clase dropout para reducir el sobreajuste durante el tiempo de entrenamiento de las neuronas.

Evitamos la aparición de los mensajes de alerta por consola.

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
```

Esta clase con unittest nos ayudara a ver que tan buena es la red neuronal que creamos con tensorflow los datos de predicción son una lista de 10 posiciones así que debe decir bien en los 10 arreglos que da como respuesta la red neuronal de la siguiente forma:

```
prediction = prediction[0]
prediction[0] => approx. 1
prediction[1] => approx. 1
...
prediction[9] => approx. 1
```

```
class TestCasePrediction(unittest.TestCase):
    def test_prediction(self):
        count, i = 0, 0
        for array in list(prediction):
            approx = array[i]
            print()
            print(array)
            print()
            print(
                approx,
                'round fixed =>', round(approx, 2),
                'porcentual =>', str(round(approx * 100, 2)) + '%'
            )
            if round(approx) == 1:
                count += 1
            i += 1
        print()
        print(
            'test accuracy =>', str(round(count / 10 * 100, 2)) + '%'
        )
        self.assertEqual(10, count)
```



Primero compilamos la red definiendo una función de pérdida y un optimizador: en nuestro caso seleccionamos categorical_crossentropy, porque tenemos múltiples categorías (como en los números 0-9). La función number_recognition_model nos crea un modelo de Sequential para la predicción de números expresados en patrones de 35 números entre 0 y 1 esta función genera tres capas la primera con 32 neuronas con función de activación "relu" otra con el doble de la anterior con 64 y también función de activación "relu" y la capa de salida con 10 neuronas con función de activación "softmax".

```
def number_recognition_model():
    # dropout = Dropout(0.20)
    layer_one = Dense(32, activation='tanh')
    layer_two = Dense(64, activation='tanh')
    layer_three = Dense(10, activation='softmax')

    number_model = Sequential()
    # number_model.add(dropout)
    number_model.add(layer_one)
    number_model.add(layer_two)
    number_model.add(layer_three)
    number_model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])

    return number_model
```

Cargamos los datos de entrenamiento para la primera versión de los datos de entrada.

```
train_data = np.loadtxt('./number_patterns_primary.txt', delimiter=',')
batch = train_data
```

```
print("data = ", train_data)
print()
```

Mostramos los datos de entrada.

Cargamos los datos de salida de la primera versión de los datos de salida.

```
target_data = np.loadtxt('./number_target_primary.txt', delimiter=',')
```



```
print("target = ", target_data)  
print()
```

Mostramos los datos de salida.

Llamar la función para tener el modelo creado con anterioridad.

```
model = number_recognition_model()
```

```
start = time.time()
```

Tomar muestra del tiempo para ver cuánto demora la ejecución en milisegundos.

Entrenamiento del modelo con variaciones en epochs y batch_size

```
model.fit(train_data, target_data, epochs=3500)
```

```
finish = time.time()  
  
print()  
print((finish - start) * 1000, 'ms o', (finish - start), 's')  
print()  
  
print('Tiempo de inicio: ', time.asctime(time.localtime(start)))  
print('Tiempo de finalización: ', time.asctime(time.localtime(finish)))  
print()
```

Muestra del tiempo en el que comenzó y finalizó

Evaluamos el modelo con sus datos de entrenamiento y verificamos la precisión de este.

```
scores = model.evaluate(train_data, target_data)  
print('\n%s: %.2f%%' % (model.metrics_names[1], scores[1] * 100))  
print()
```



```
predict_data = model.predict(train_data)
print(predict_data)
print()
```

Hacemos una predicción con los datos generales de entrada.

Ingresamos datos de test para el número 6 para luego verificar el resultado.

```
test_data = np.loadtxt('./testing_data_primary.txt', delimiter=',')
print(test_data)
```

```
prediction = model.predict(test_data)
print("predicted softmax output => ", prediction)
```

Hacemos una predicción con los datos de test.

Salvamos el modelo para poder volver a utilizar nuestra red neuronal.

```
model.save('numbers_model_primary.h5')

if __name__ == '__main__':
    unittest.main()
```

```
from keras.models import load_model
```

Cargamos el modelo con la función load_model del módulo de "models" de keras.

Cargamos los datos de testing.

```
test_data = np.loadtxt('./testing_data_primary.txt', delimiter=',')
print(test_data)

view_data = test_data.reshape((10, 7, 5))

pl.gray()

for array in list(view_data):
    pl.matshow(array)

pl.show()
```



LOS LENGUAJES UTILIZADOS PARA EL DESARROLLO DE LA APLICACIÓN WEB:

| LENGUAJE DE PROGRAMACIÓN | ÍCONO | VERSIÓN | EXPLICACIÓN |
|--------------------------|-------|-----------------|--|
| Python | | 3.9.5 | Utilizado tanto para el desarrollo como despliegue de la aplicación. |
| JavaScript | | ECMAScript 2015 | Utilizado para la interactividad de la aplicación WEB. |

CODIFICACIÓN EN JAVASCRIPT:

| SUBCARPETA | ARCHIVO |
|--------------|----------|
| static → js | index.js |

Objeto de referencia para entrenar el modelo de Keras.

```
let arr = [
    0, 0, 0, 0, 0,
    0, 0, 0, 0, 0,
    0, 0, 0, 0, 0,
    0, 0, 0, 0, 0,
    0, 0, 0, 0, 0,
    0, 0, 0, 0, 0,
    0, 0, 0, 0, 0
];
```

```
const toggleArray = (id) => {
    id = Number(id);

    const $box = d.getElementById(id);

    (arr[id - 1] === 1)
        ? arr[id - 1] = 0
        : arr[id - 1] = 1;

    handleClass($box, 'active')
}
```

Función que verifica si el elemento del DOM con ese id en su lista de clases contiene '. active' en caso no lo tenga le agrega a su lista de clases y le asigna al arreglo arr = [] en la posición indicada el valor uno si no le asigna el valor 0 y le quita la clase 'active'.



Función asíncrona que carga el modelo ubicado en la ruta /static/neuronal-network/, y con el objeto "tf" su función loadLayersModel con el parámetro de la ruta nos reconstruye el modelo entrenado en Python.

```
const sendData = async () => {
    handleClass($modal, 'is-open');
    const model = await tf.loadLayersModel('..../static/neuronal-network/model.json');
    return await model.predict(tf.tensor([arr]));
}
```

```
w.addEventListener('click', async (e) => {
    if (e.target.matches('.box')) {
        toggleArray(e.target.id)
    }
})
```

Listener para la escucha de eventos en nuestra aplicación con el evento del "click" de esta manera podemos escuchar todos los "clicks" que se hagan con la delegación de eventos

Listeners para los botones de los modales y otros.

```
$btnClose.addEventListener('click', () => {
    handleClass($modal, 'is-open')
})

$modal.addEventListener('click', () => {
    handleClass($modal, 'is-open');
})

$container.addEventListener('click', (e) => {
    handleCloseStop(e);
})

$btnClear.addEventListener('click', () => {
    clearBoxes();
})
```



LOS LENGUAJES UTILIZADOS PARA CREAR LA INTERFAZ WEB:

| LENGUAJES CREAR LA INTERFAZ WEB | ÍCONO | VERSIÓN | EXPLICACIÓN |
|--|-------|---------|--|
| HTML | | HTML 5 | Utilizamos este lenguaje de marcado, porque es el único que se reconoce en los navegadores modernos. |
| CSS | | CSS 3 | Utilizamos este lenguaje de estilos en cascada para el diseño gráfico y la presentación de nuestro archivo HTML. |

CODIFICACIÓN EN HTML:

| SUBCARPETA | ARCHIVO |
|------------|------------|
| templates | index.html |

Head:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="{{ url_for('static', filename='css/index.css') }}"/>
    </head>
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@100;300;400;500&display=swap" rel="stylesheet">
    <title>Number Recognition App</title>
  </head>
```

En la parte inicial del código de html empezamos con el lenguaje "en" que significa que esta en español y luego en el "head" introducimos un link que hace referencia a la hoja de estilos que se utiliza en su respectiva ruta de acceso en la que se encuentra y asimismo con los Fonts utilizados de

Body: Header

```
<body>
  <header>
    <h1 class="title">Number Recognition App</h1>
  </header>
```

En esta parte del body mediante la etiqueta header y h1 se introduce el nombre del TÍTULO de nuestra aplicación en la web.



Main:

```
<main class="main">
  <div class="box" id="1"></div>
  <div class="box" id="2"></div>
  <div class="box" id="3"></div>
  <div class="box" id="4"></div>
  <div class="box" id="5"></div>
  <div class="box" id="6"></div>
  <div class="box" id="7"></div>
  <div class="box" id="8"></div>
  <div class="box" id="9"></div>
  <div class="box" id="10"></div>
  <div class="box" id="11"></div>
  <div class="box" id="12"></div>
  <div class="box" id="13"></div>
```

```
<div class="box" id="14"></div>
<div class="box" id="15"></div>
<div class="box" id="16"></div>
<div class="box" id="17"></div>
<div class="box" id="18"></div>
<div class="box" id="19"></div>
<div class="box" id="20"></div>
<div class="box" id="21"></div>
<div class="box" id="22"></div>
<div class="box" id="23"></div>
<div class="box" id="24"></div>
<div class="box" id="25"></div>
<div class="box" id="26"></div>
```

```
<div class="box" id="27"></div>
<div class="box" id="28"></div>
<div class="box" id="29"></div>
<div class="box" id="30"></div>
<div class="box" id="31"></div>
<div class="box" id="32"></div>
<div class="box" id="33"></div>
<div class="box" id="34"></div>
<div class="box" id="35"></div>
</main>
```

Se pone una class de nombre main para luego ser utilizado en la hoja de estilos css y así añadir las dimensiones y características que se necesitan, asimismo se creó divisiones a las que se le puso clases de nombre "box" y cada uno con un "id" enumerado del 1 al 35, los cuales son los cuadros mostrados en la interfaz que son 35 en total debido a que guardan relación con el arreglo de 7 filas x 5 columnas.

Button:

```
<button class="btn__send">Enviar Datos</button>
<button class="btn__clear">
  <p>Limpiar</p>
  
</button>
```

Usamos la metodología BEM para la especificación de las clases especiales que tendrán interactividad para el usuario.

El motor de plantillas de *jinja* reconoce un parámetro expresado en un cerrado de doble llave para especificar rutas de archivos que existan en el directorio principal de la aplicación.

Footer:

```
<footer>
  <ul class="list__details">
    <li>
      <a href="https://www.facebook.com/jochizan" target="_blank">
        
      </a>
      <p>facebook</p>
    </li>
```

```
<li>
  <a
    href="https://github.com/Jochizan/numbers-recognition-app"
    target="_blank"
  >
    
  </a>
  <p>github</p>
</li>
```

```
<li>
  <a href="https://www.python.org/" target="_blank">
    
  </a>
  <p>python</p>
```

En el footer de nuestra aplicación web se utilizan las etiquetas para ordenar los iconos que permitirán tanto el acceso mediante "a href" donde se encuentra el link de enlace al que conduce y en "img" especifica la ruta donde se encuentran los iconos.



Scripts:

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@3.0.0/dist/tf.min.js"></script>
```



Este script trae el CDN oficial de tensorflow para el lenguaje javascript y con este podremos cargar el modelo entrenado en Python, pero convertido para el soporte de nuestra red neuronal en javascript.

```
<script src="{{ url_for('static', filename='js/index.js') }}"></script>
```



Este script nos trae el archivo principal javascript para la interactividad de la aplicación web.

TECNOLOGÍAS PARA EL SEGUIMIENTO DE CAMBIOS DENTRO DE LA APLICACIÓN WEB COMO LA RED NEURONAL:

| TECNOLOGÍAS | ÍCONO | EXPLICACIÓN |
|-------------|-------|--|
| GitHub | | Es una de las versiones para la mantenibilidad del software en equipos de desarrolladores a través de la web ya que nos provee de un repositorio remoto vinculado a un repositorio local de Git. |
| Git | | Es un controlador de versiones avanzado para proyectos en cualquier lenguaje de programación (el más usado por la comunidad de desarrolladores de software). |

REPOSITORIOS DE NUESTRO PROYECTO:

| REPOSITORIOS | LINK DE ENLACE |
|----------------------------------|---|
| REPOSITORIO DE LA APLICACIÓN WEB | https://github.com/Jochizan/numbers-recognition-app |
| REPOSITORIO DE LA RED NEURONAL | https://github.com/Jochizan/maching-learning/tree/master/src/recognition_number |

LINK DE LA APLICACIÓN DESPLEGADA EN LA WEB:

<https://number-recognition-app.herokuapp.com/>



RESULTADOS DE LA APLICACIÓN WEB CON LA RED NEURONAL:

Se muestra como resultado en la aplicación web que con la ayuda y el entrenamiento de la red neuronal después de hacer pruebas con diferentes funciones de activación como sigmoide,tangencial,rule,softmax entre otros, observamos que los mejores datos con mayor probabilidad lo obtuvimos con las funciones de activación tanh y relu en las capas ocultas, y de salida con softmax; es así que con esa función de activación se procedió a evaluar en nuestra aplicación web debido a que también permitía el reconocimiento de los números en distintas formas que es lo que se buscaba en mayor probabilidad para relacionar la información con nuestra interfaz web y así realizar las pruebas adecuadas que mostraremos a continuación:

1.Se muestra la interfaz principal donde se aprecia la aplicación web con flask sencilla, importando flask para generar un sencillo servidor en el puerto 5000 del localhost por defecto. Se implementó una tabla de 5 columnas x 7 filas para la colocación de los números, un botón “Enviar Datos” para el reconocimiento del número implementado, un botón “Limpiar” para devolver a su estado actual la tabla y por último se implementó iconos incluyendo imágenes jpg los cuales realizan diferentes acciones cada una.

The screenshot shows a web application titled "Number Recognition App". At the top, there is a 5x7 grid of input fields for entering numbers. Below the grid is a blue button labeled "Enviar Datos". Underneath the button is a "Limpiar" button with a trash can icon. At the bottom of the page, there are four social media and technology icons: Facebook, GitHub, Python, and TensorFlow. The background has a blue-to-yellow gradient.

Number Recognition App

Enviar Datos

Limpiar

facebook

github

python

tensorflow

Iconos diseñados por Pixel perfect from www.flaticon.es

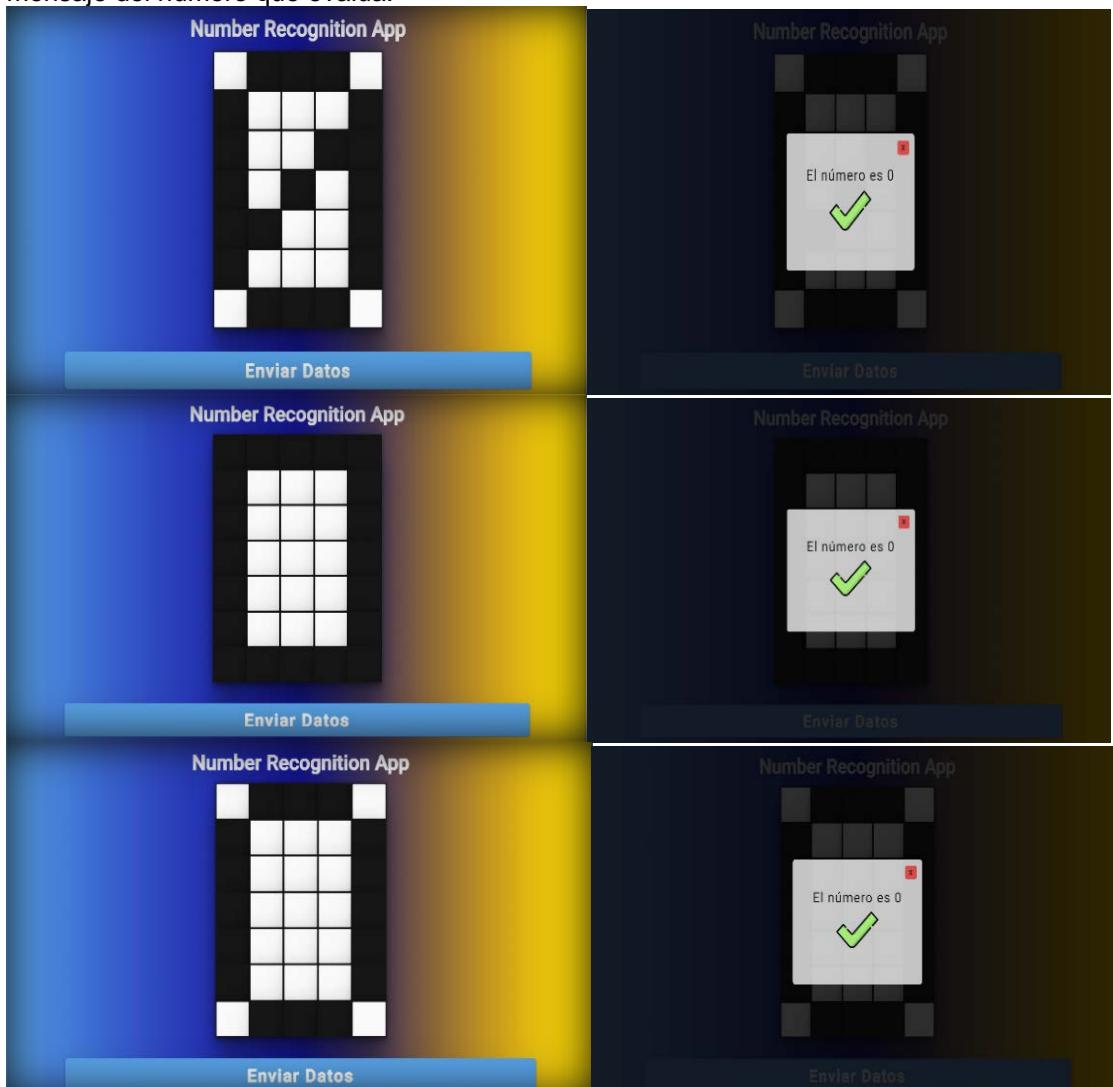
sitio diseñado por jochizan



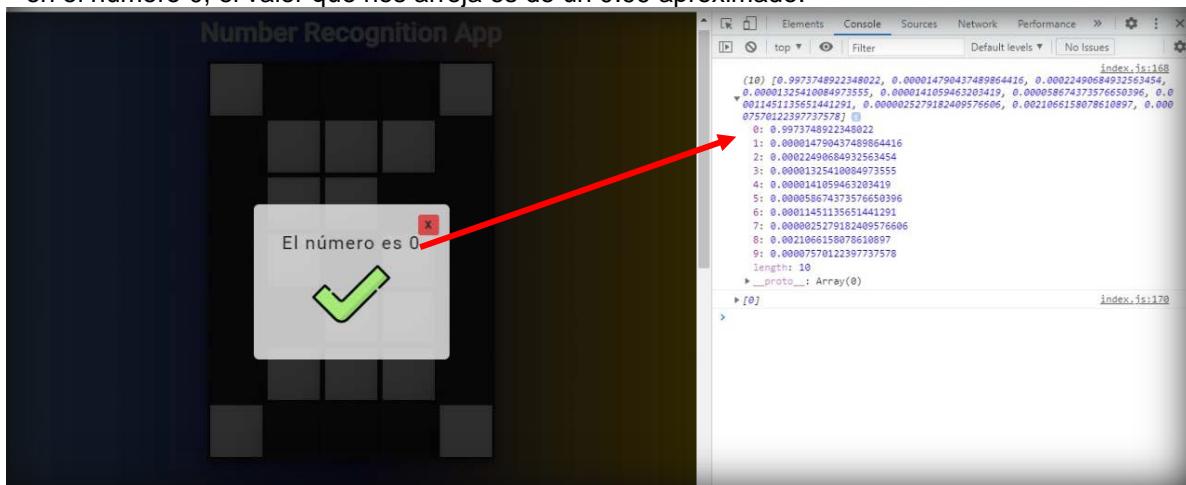
UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ FACULTAD DE INGENIERÍA DE SISTEMAS



2. Se hace la prueba haciendo click ingresando el número 0 en la tabla en distintos patrones, obteniendo como resultado la respuesta en pantalla donde se reconoce el número y muestra el mensaje del número que evalúa.



- Se muestra por consola que a la hora de validar el patrón ingresado donde se observa como en el número 0, el valor que nos arroja es de un 0.99 aproximado.



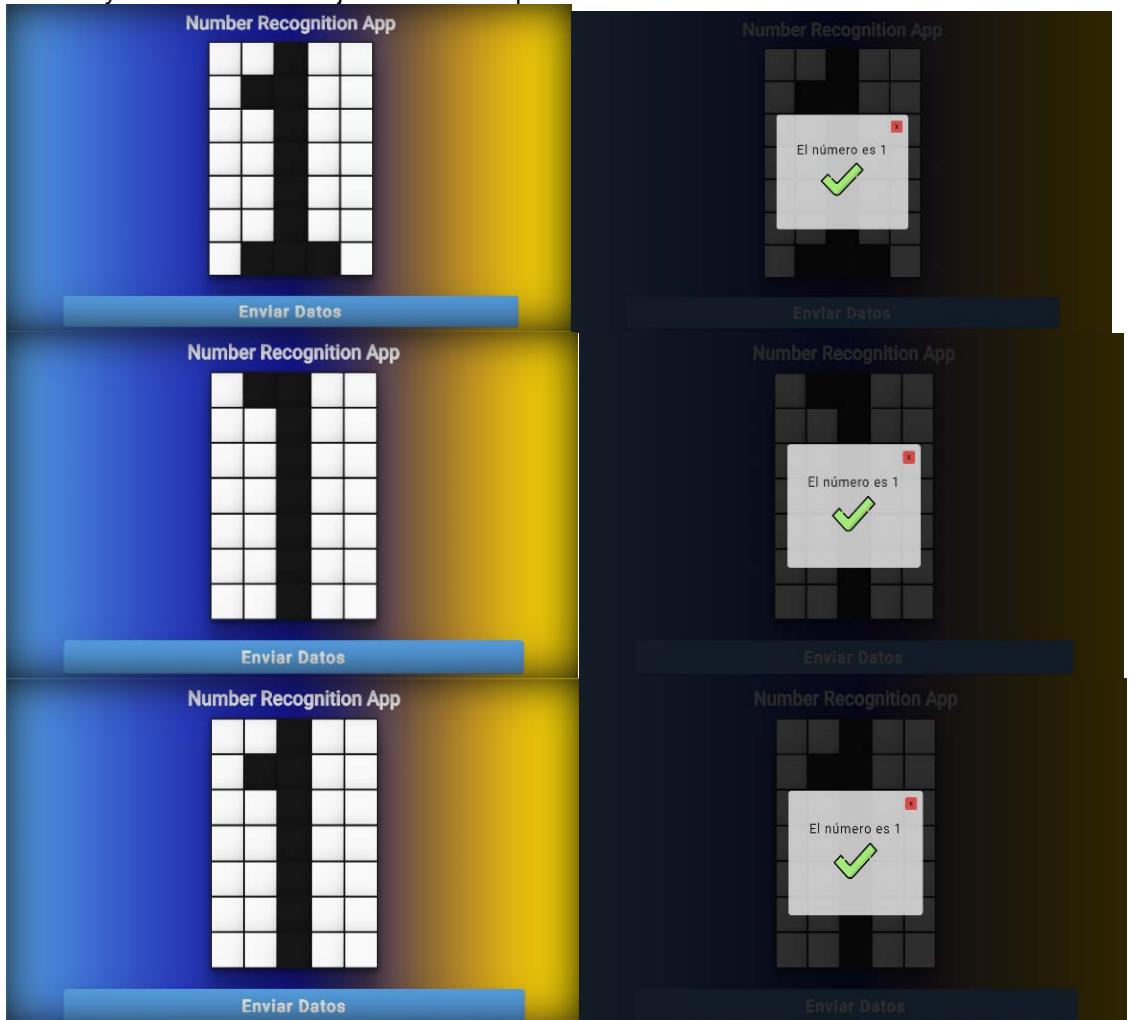


UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

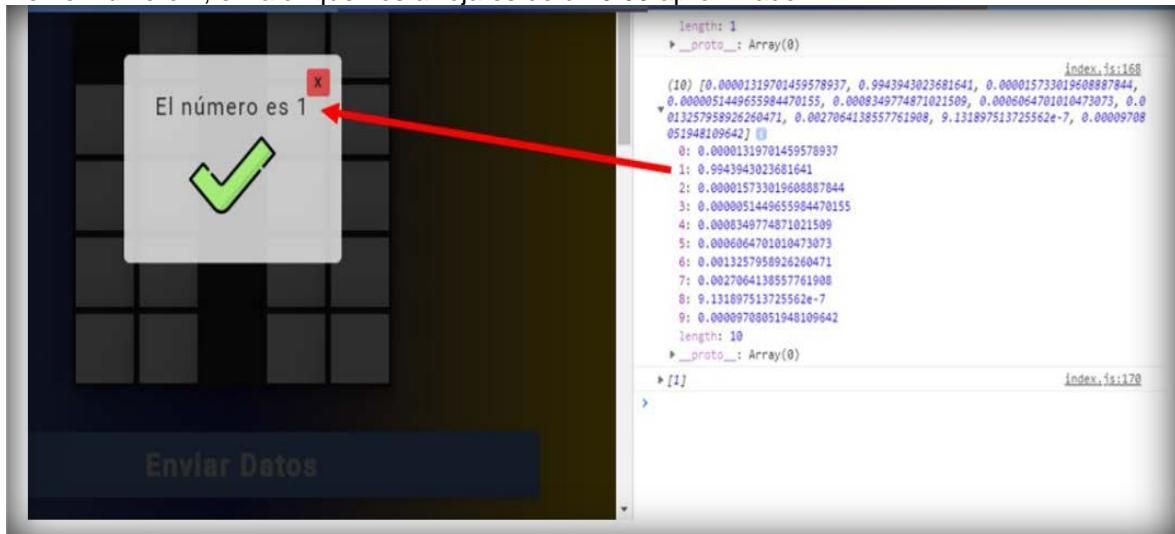
FACULTAD DE INGENIERÍA DE SISTEMAS



3. Se hace la prueba haciendo click en los recuadros ingresando la forma del número 1 en distintos patrones, obteniendo como resultado la respuesta en pantalla donde se reconoce el número y muestra el mensaje del número que evalúa.



- Se muestra por consola que a la hora de validar el patrón ingresado donde se observa como en el número 1, el valor que nos arroja es de un 0.99 aproximado.



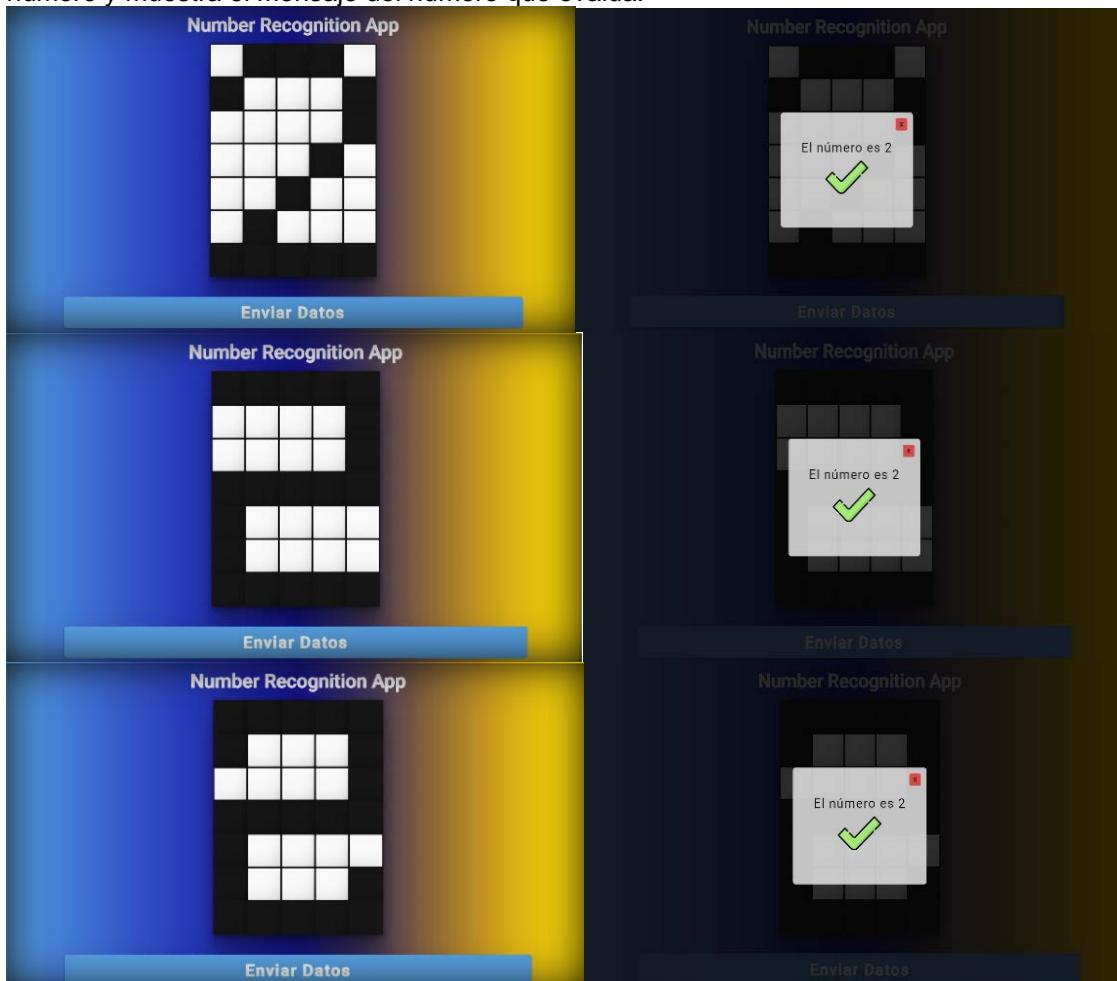


UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

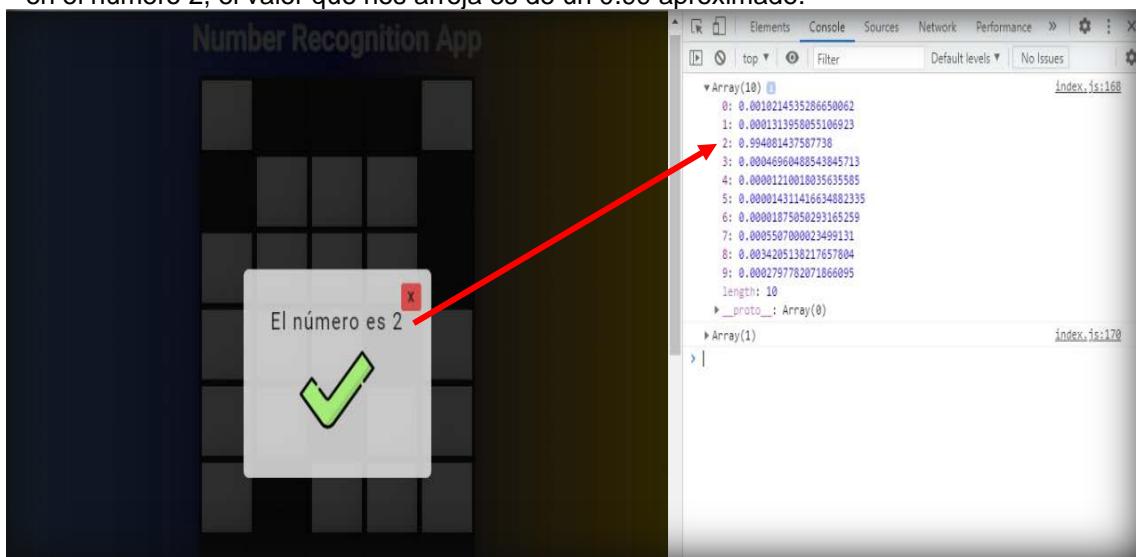
FACULTAD DE INGENIERÍA DE SISTEMAS



4. Se hace la prueba haciendo click en los recuadros ingresando la forma del número 2 en distintos patrones, obteniendo como resultado la respuesta en pantalla donde se reconoce el número y muestra el mensaje del número que evalúa.



- Se muestra por consola que a la hora de validar el patrón ingresado donde se observa como en el número 2, el valor que nos arroja es de un 0.99 aproximado.



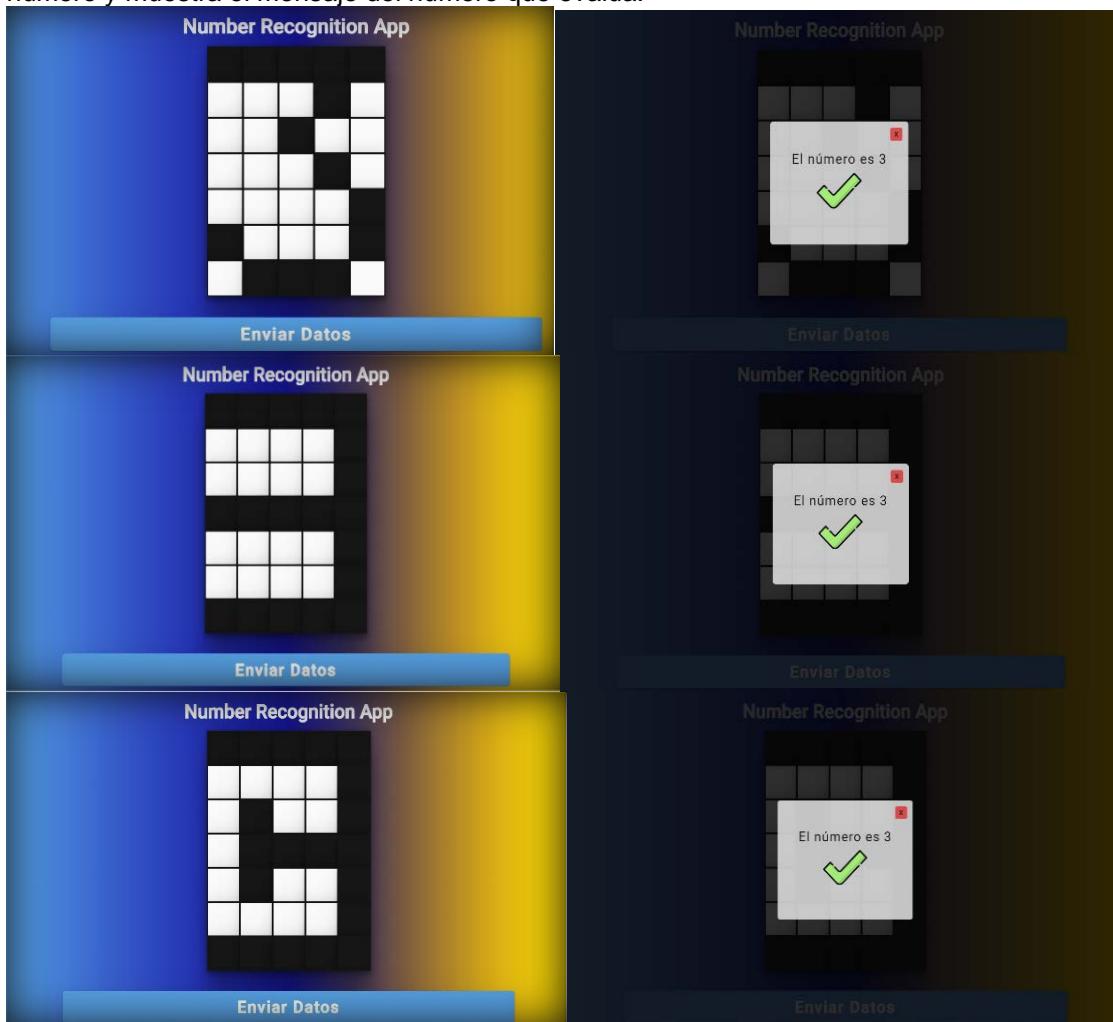


UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

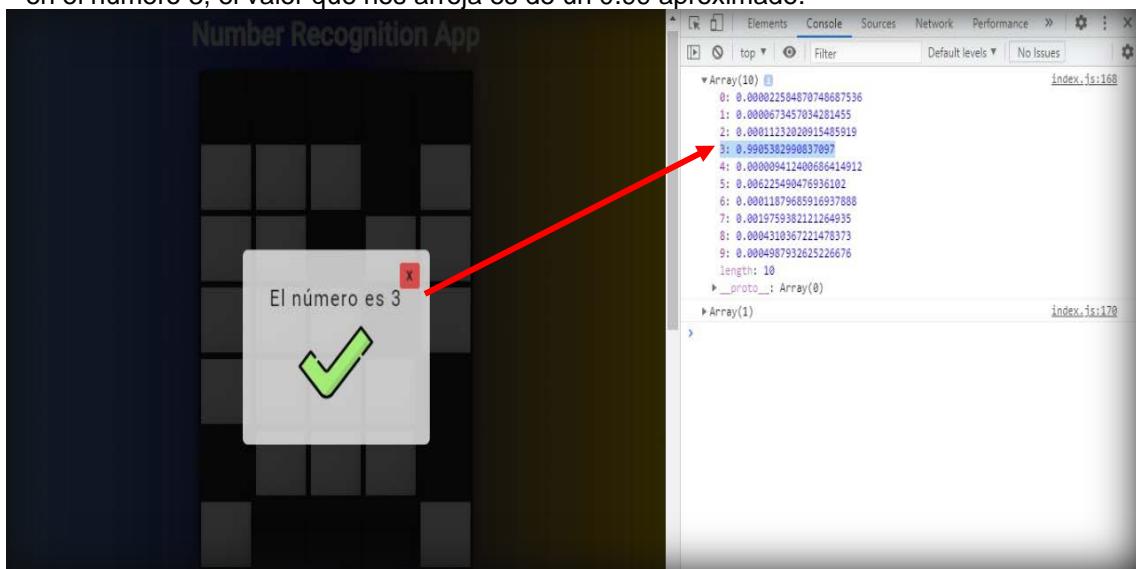
FACULTAD DE INGENIERÍA DE SISTEMAS



5. Se hace la prueba haciendo click en los recuadros ingresando la forma del número 3 en distintos patrones, obteniendo como resultado la respuesta en pantalla donde se reconoce el número y muestra el mensaje del número que evalúa.



- Se muestra por consola que a la hora de validar el patrón ingresado donde se observa como en el número 3, el valor que nos arroja es de un 0.99 aproximado.



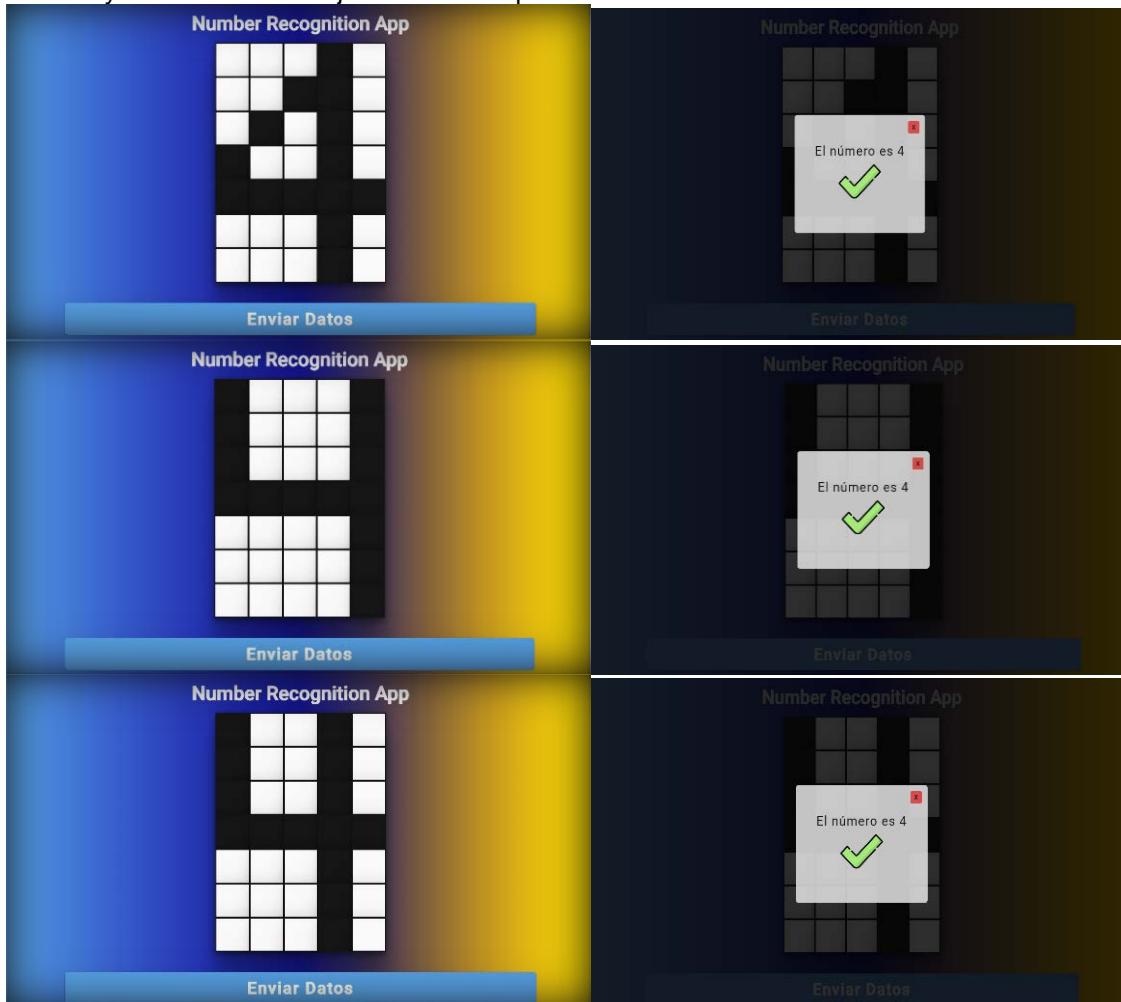


UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

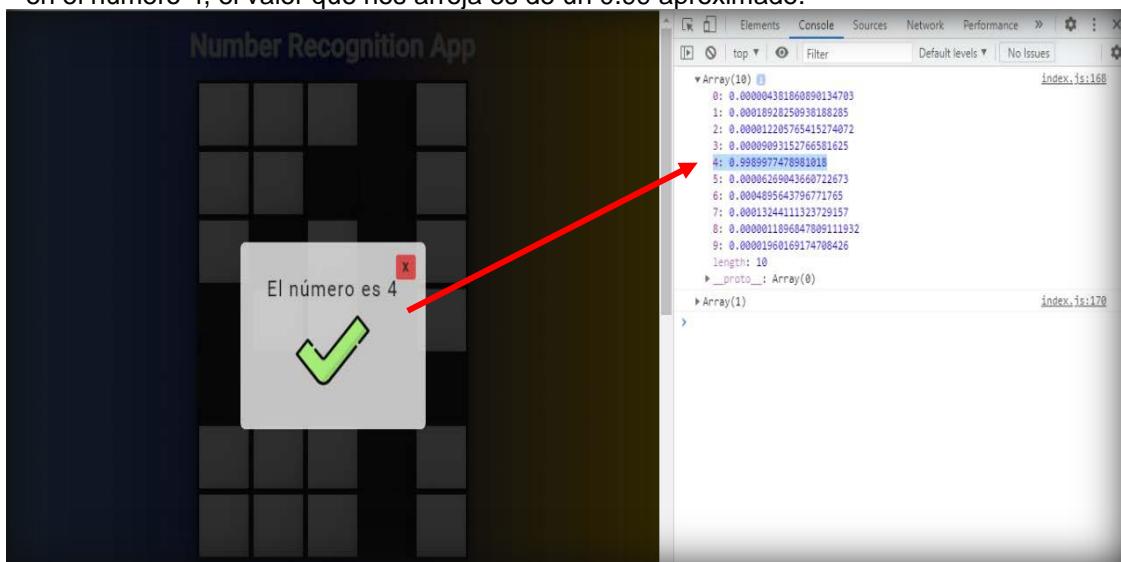
FACULTAD DE INGENIERÍA DE SISTEMAS



5. Se hace la prueba haciendo click en los recuadros ingresando la forma del número 4 en distintos patrones, obteniendo como resultado la respuesta en pantalla donde se reconoce el número y muestra el mensaje del número que evalúa.



- Se muestra por consola que a la hora de validar el patrón ingresado donde se observa como en el número 4, el valor que nos arroja es de un 0.99 aproximado.



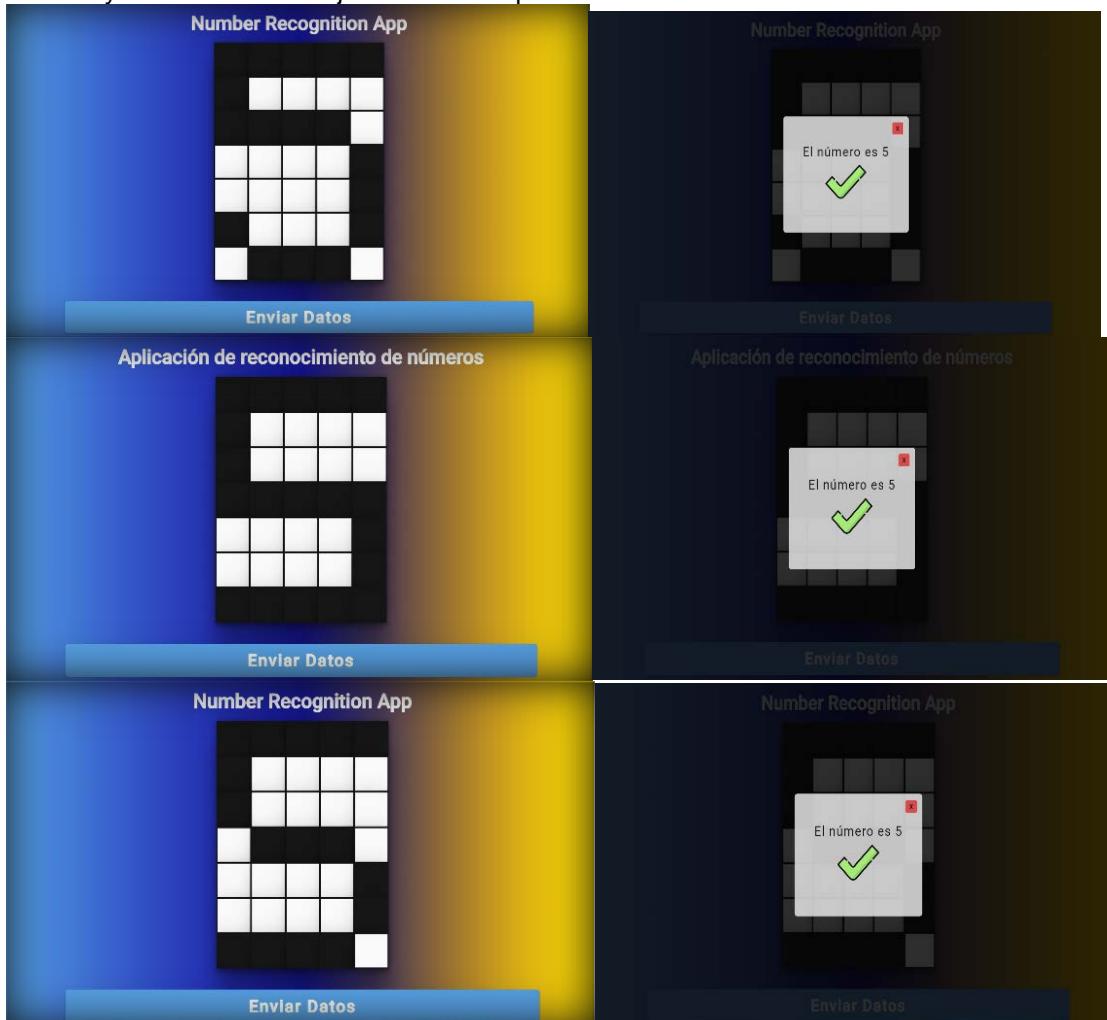


UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

FACULTAD DE INGENIERÍA DE SISTEMAS



6. Se hace la prueba haciendo click en los recuadros ingresando la forma del número 5 en distintos patrones, obteniendo como resultado la respuesta en pantalla donde se reconoce el número y muestra el mensaje del número que evalúa.



- Se muestra por consola que a la hora de validar el patrón ingresado donde se observa como en el número 5, el valor que nos arroja es de un 0.99 aproximado.

A screenshot of the "Number Recognition App" interface. On the left, a 4x4 grid shows a pattern of white squares forming the number 5. A small window to the right displays "El número es 5" with a green checkmark. A red arrow points from the console output area to this window. On the right, a browser's developer tools console tab is open, showing the following array of values:

```
(10) [0.002558915975852108, 0.0001784133055480197, 0.00001755342395881598, 0.000999588685807586, 0.0001175618965351351, 0.9951956907707214, 0.0001291861995235085, 0.0000467491453397088, 0.00005616854105028324, 0.000050154787674546]
```

The value 0.9951956907707214 is highlighted with a red arrow. The console also shows other numerical values and array details.

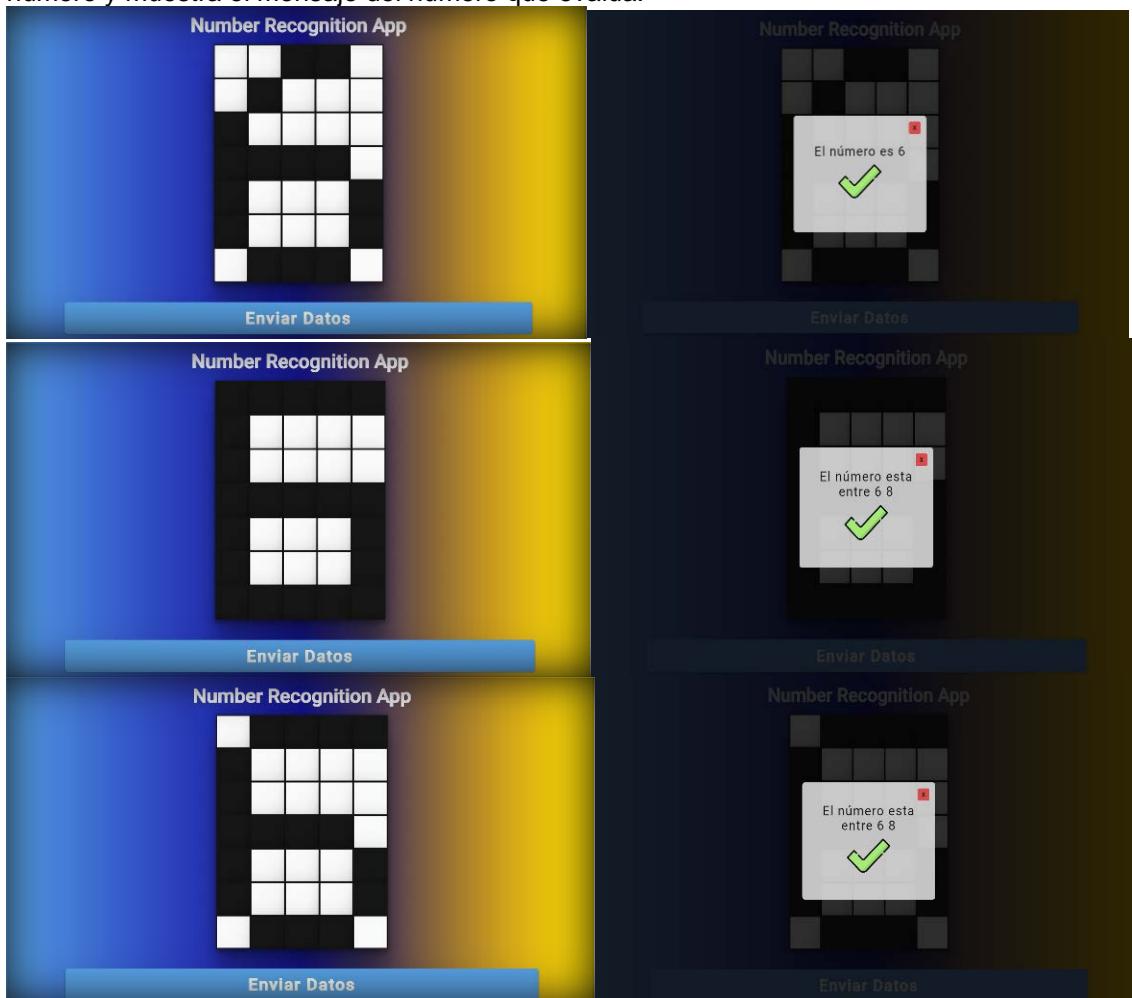


UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ

FACULTAD DE INGENIERÍA DE SISTEMAS



7. Se hace la prueba haciendo click en los recuadros ingresando la forma del número 6 en distintos patrones, obteniendo como resultado la respuesta en pantalla donde se reconoce el número y muestra el mensaje del número que evalúa.



- Se muestra por consola que a la hora de validar el patrón ingresado donde se observa como en el número 6, el valor que nos arroja es de un 0.99 aproximado.

```
index.js:168
(10) [0.000179140872056076, 0.00018215170712210238, 0.0001284471727558411
, 0.00003796358942054212, 0.0000558073526169392, 0.00461480842047645,
0.9984640574455261, 0.000007300310244318098, 0.010579891502857208, 0.000009
3820544861955577] ⓘ
0: 0.000179140872056076
1: 0.00018215170712210238
2: 0.0001284471727558411
3: 0.00003796358942054212
4: 0.0000558073526169392
5: 0.9984640574455261
6: 0.9984640574455261
7: 0.000007300310244318098
8: 0.010579891502857208
9: 0.0000093820544861955577
length: 10
▶ _proto__: Array(0)
> [6]
```



UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ FACULTAD DE INGENIERÍA DE SISTEMAS



8. Se hace la prueba haciendo click en los recuadros ingresando la forma del número 7 en distintos patrones, obteniendo como resultado la respuesta en pantalla donde se reconoce el número y muestra el mensaje del número que evalúa.

The figure consists of four screenshots of a mobile application titled "Number Recognition App". Each screenshot shows a 4x8 grid of squares. In each grid, some squares are black and some are white, forming the shape of the number 7. Below each grid is a blue button labeled "Enviar Datos". To the right of each grid, a small pop-up window displays the text "El número es 7" next to a green checkmark icon. The backgrounds of the screenshots are blue and yellow gradients.

- Se muestra por consola que a la hora de validar el patrón ingresado donde se observa como en el número 7, el valor que nos arroja es de un 0.99 aproximado.

The figure shows a screenshot of a web browser's developer tools. The "Console" tab is selected, displaying an array of 18 numerical values. A red arrow points from the text "El número es 7" in the app's UI to the value at index 7 of the array, which is highlighted in blue. The array values are as follows:

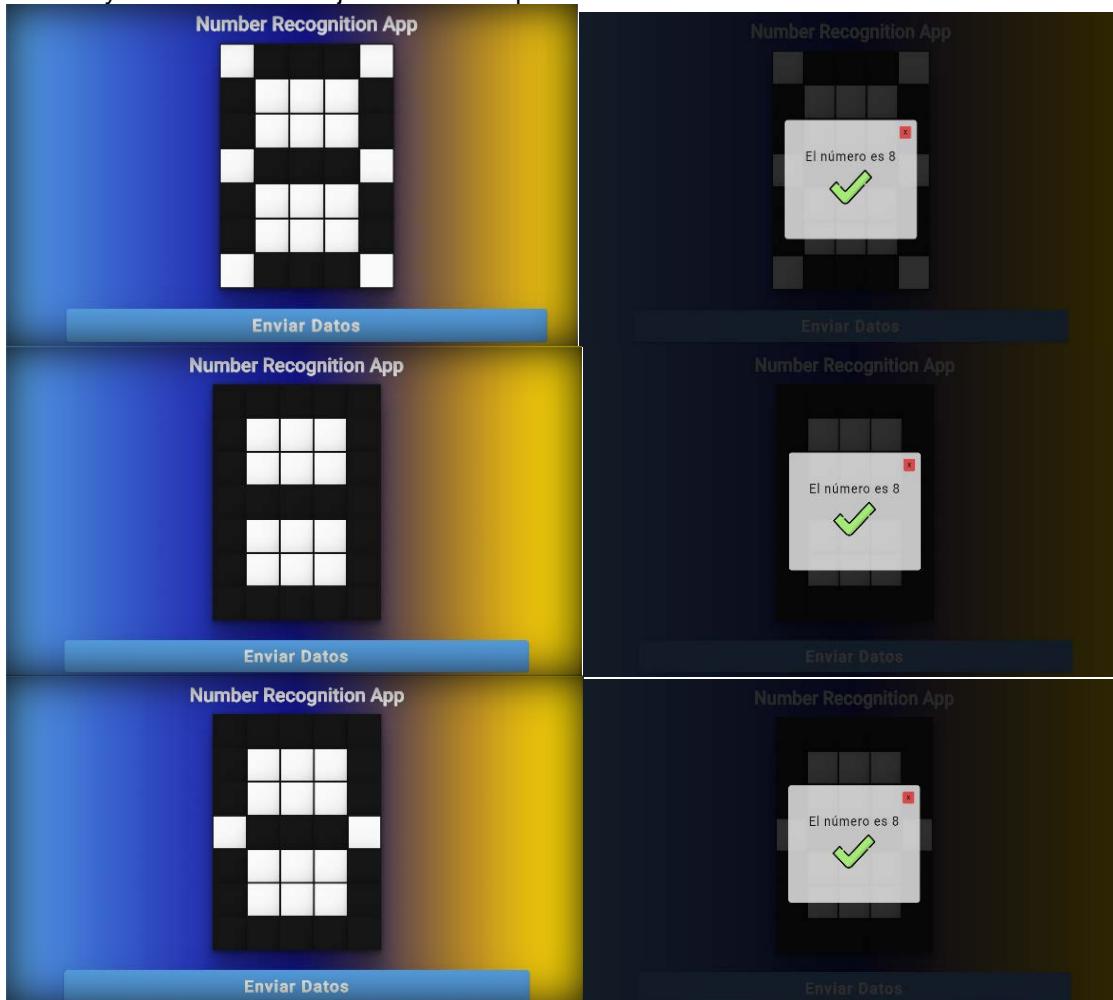
```
index.js:168
Array(18)
0: 0.00015636409807484597
1: 0.001258923942905736
2: 0.00135823292056286335
3: 0.0013524845708161592
4: 0.0000375780735826984
5: 0.00009662129235906377
6: 0.000004426533218473196
7: 0.5947875738143921
8: 0.00005279018477288678
9: 0.0003604257944971323
length: 10
__proto__: Array(0)
index.js:170
```



UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ FACULTAD DE INGENIERÍA DE SISTEMAS



9. Se hace la prueba haciendo click en los recuadros ingresando la forma del número 8 en distintos patrones, obteniendo como resultado la respuesta en pantalla donde se reconoce el número y muestra el mensaje del número que evalúa.



- Se muestra por consola que a la hora de validar el patrón ingresado donde se observa como en el número 8, el valor que nos arroja es de un 0.99 aproximado.





UNIVERSIDAD NACIONAL DEL CENTRO DEL PERÚ FACULTAD DE INGENIERÍA DE SISTEMAS



10. Se hace la prueba haciendo click en los recuadros ingresando la forma del número 9 en distintos patrones, obteniendo como resultado la respuesta en pantalla donde se reconoce el número y muestra el mensaje del número que evalúa.

The figure consists of four screenshots of a mobile application titled "Number Recognition App". Each screenshot shows a 4x4 grid of squares. In the first two screenshots, the grid contains a pattern of white squares forming the number '9'. In the third and fourth screenshots, the pattern is rotated 90 degrees clockwise. Below each grid is a blue button labeled "Enviar Datos". To the right of each screenshot is a smaller window showing the app's response: a message box with the text "El número es 9" and a green checkmark icon.

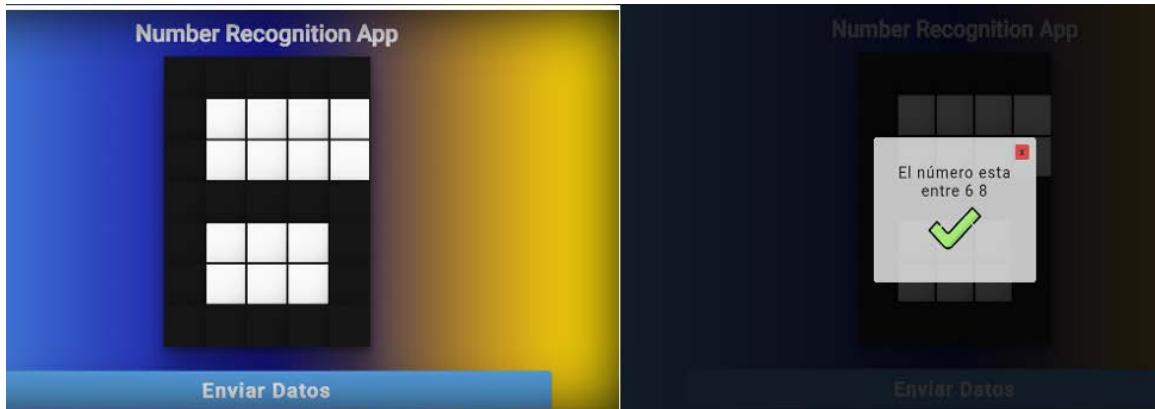
- Se muestra por consola que a la hora de validar el patrón ingresado donde se observa como en el número 9, el valor que nos arroja es de un 0.99 aproximado.

This screenshot shows the "Number Recognition App" running on a device. A red arrow points from the bottom of the app screen to the developer tools' "Console" tab in the browser's developer toolbar. The console output shows an array of 10 numerical values, with the 9th value highlighted in red, indicating it is approximately 0.99.

```
Array(10) [0: 0.00003584771911846474, 1: 0.00001852652619776968, 2: 0.000018677126718102954, 3: 0.00023349876573774964, 4: 0.00040359056889211, 5: 0.000381568621378392, 6: 0.00000441098154624342, 7: 0.00017110964108724147, 8: 0.004511130508035421, 9: 0.9942216277122498] index.js:168
```



LIMITACIONES:



- Encontramos que en la red neuronal utilizada en nuestra aplicación web puede fallar en la predicción de algunos números como en los patrones del 6 y el 8 en una forma debido a que los recuadros evaluados en forma son casi los mismo causando resultado tanto como 6 y 8, como se muestra en la imagen.

CONCLUSIONES

- Realizando las distintas pruebas para hallar la mejor red neuronal a utilizar en la aplicación web nos permitió escoger la más adecuada que nos arrojaba mejores resultados 9/10, utilizando así las funciones de activación tanh y relu en las capas ocultas y en las de salida softmax, obteniendo así la mejor predicción en los distintos patrones de cada número a evaluar.
- Concluimos que crear una interfaz es sencilla con ayuda del framework flask y el motor de plantillas jinja.
- Usando la función de abandono “Dropout” de la API de Keras nos ayudo a establecer redes neuronales sin un sobreajuste.
- La importancia del proyecto radica en que la aplicación web presenta una interfaz adecuada y comprensible para evaluar mediante patrones el reconocimiento de números del 0 al 9 aplicando la inteligencia artificial mediante el uso y entrenamiento por medio de las redes neuronales.



Bibliografía

- BIGDATA. (2020). *Optimización de descenso de gradiente de Code Adam desde cero*. Obtenido de <https://topbigdata.es/optimizacion-de-descenso-de-gradiente-de-code-adam-desde-cero/>
- Buhigas, J. (14 de Febrero de 2018). *Todo lo que necesitas saber sobre TensorFlow, la plataforma para Inteligencia Artificial de Google*. Obtenido de <https://puentesdigitales.com/2018/02/14/todo-lo-que-necesitas-saber-sobre-tensorflow-la-plataforma-para-inteligencia-artificial-de-google/>
- Calvo, D. (7 de Diciembre de 2018). *Función de activación – Redes neuronales*. Obtenido de <https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/>
- Duran, A. G. (2018). *¿Que es Flask?* Obtenido de <https://openwebinars.net/blog/que-es-flask/>
- ENI. (2017). *Inteligencia artificial fácil - Machine Learning y Deep Learning prácticos*. Obtenido de <https://www.ediciones-eni.com/open/mediabook.aspx?idR=8dd2ca32769cb24b49648b15ef8e777e>
- Telefonica DATA UNIT. (2018). *Función de Activación*. Obtenido de <https://luca-d3.com/es/data-speaks/diccionario-tecnologico/funcion-activacion>
- Utrera, J. (20 de Junio de 2018). *Deep Learning básico con Keras* . Obtenido de <https://enmilocalfunciona.io/deep-learning-basico-con-keras-parte-1/>