Congr. Int. en Ing. Electrónica. Mem. ELECTRO, Vol. 43 pp. 44 - 49, Oct 2021, Chihuahua, Chih. México
http:// electro.itchihuahua.edu.mx/revista
ISSN 1405-2172

Sistema de clasificación de agua jabonosa empleando Redes Neuronales

Luis Alberto Reyes Ibarra, Perla Alejandra Herrera Castillo, Sergio Alberto Sifuentes Montelongo, Raúl Villanueva Vallejo

Universidad Politécnica de Durango Carr. Dgo-Mex Km 9.5 S/N Loc. Dolores Hidalgo
C.P. 34300 Durango, Dgo Tel. 618 1501323
Ing. Telemática
luis.reyes@unipolidgo.edu.mx, perla.herrera@unipolidgo.edu.mx, sergio.sifuentes@unipolidgo.edu.mx, raul.villanueva@unipolidgo.edu.mx

RESUMEN

El prototipo presentado forma parte de un proyecto global que pretende ser un instrumento en casas habitación que cuenten con una lavadora, ya que este toma como entrada el agua de enjuague de lavadoras convencional o automática. El objetivo es tratar el agua de enjuague o jabonosa para que se le pueda ser reutilizada en usos comunes en una casa. El sistema de tratamiento utiliza diversos componentes como un agitador, inyector de minerales, filtro adaptado para las características del agua, etapa de sensado y principalmente un algoritmo inteligente con Redes Neuronales que permite clasificar el uso recomendado del agua una vez tratada. Este algoritmo es implementado en una Raspberry Pi 4 en conjunto con una etapa de sensores de pH y turbidez.

La principal ventaja de este prototipo es que utiliza un modelo de red neuronal multicapa Perceptrón que aporta certeza en la determinación del uso del agua. La automatización del sistema coadyuva al uso efectivo del agua.

Además se busca reducir el consumo de agua el cual es excesivo debido a que en promedio se consumen 200 litros en ciclos de lavados, esos litros pueden ser reutilizados haciendo el tratamiento adecuado mediante el uso de este prototipo.

Palabras clave: Tratamiento de Agua, Raspberry Pi 4, Python, Ph, Turbidez, Redes neuronales

ABSTRACT

The presented algorithm is part of a global project that pretends to be an instrument in residential houses that have a washing machine, since the designed prototype takes the input water from conventional or automatic washing machines as input. The objective is to treat the water of soap to can use it to various common uses in a house. The treatment system uses various components : an agitator, a mineral injector, a filter adapted to the characteristics of the water, a sensing stage and, a mainly, an intelligent algorithm with Neural Networks that allows classifying the use recommended of the water once it has been treated. This algorithm is implemented on a Raspberry Pi 4 in conjunction with a stage of pH and turbidity sensors.

The main advantage of this prototype is that it uses a multilayer Perceptron neural network model that provides certainty in determining the use of water. The automation of the system contributes to the effective use of water.

In addition, it seeks to reduce water consumption which is excessive due to the fact that on average 200 liters are consumed in washing cycles, those liters can be reused making the appropriate treatment through the use of this prototype.

Keywords: Treated water, Raspberry Pi 4, Python, Ph, Turbidity

1. INTRODUCCIÓN

Según el INEGI en México un 70.9% de los hogares utiliza una lavadora manual o automática, por ello, este proyecto pretende desarrollar un sistema que permita la reutilización del agua de los ciclos de lavado aplicando filtros y haciendo uso de algoritmos inteligentes para la toma de decisiones de acuerdo al tratamiento del agua requerido [1].

El prototipo propuesto pretende contribuir en dos aspectos primordiales: 1. Uso de tecnologías de impacto lo que se logra usando redes neuronales, sensores de pH y turbidez que permiten contar con un sistema inteligente que proporciona agua tratada para diversos usos. 2. Ahorro en el consumo de agua en casas habitación.

Una de las ventajas de este proyecto es que no existen máquinas automatizadas para tratar el agua de los ciclos de lavado de una lavadora convencional o automática, actualmente existen filtros comerciales pero éstos no monitorean la calidad del agua lo que no permite el uso adecuado.

En este prototipo el sistema indica el uso que le puede dar el usuario al agua una vez tratada, para esto se realizan mediciones de los niveles de turbidez y pH, éste procedimiento se realiza empleando redes neurales que toman decisiones para asignar etiquetas para recomendar el uso del agua, dichas etiquetas tienen un patrón de variación entre calidad óptima, regular y tratar (esta clase se refiere a que debe realizar un segundo ciclo de tratamiento).

El principal valor agregado del proyecto es el de consumir menos agua por hogar ante la problemática mundial del uso desmedido del agua.

El siguiente desarrollo está orientado a mostrar la obtención de un modelo de red neuronal que permite clasificar el agua que es tratada mediante procesos de floculación y coagulación. El modelo de red que se presenta es similar al mostrado en la figura 1, se trata de un modelo multicapa Perceptrón (MLP) con 2 características de entrada y tres salidas.

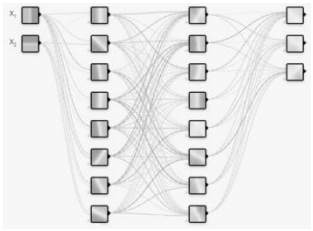


Figura 1. Modelo de Red neuronal multicapa.

1.1 Tecnologías para el reciclaje de las aguas grises

Existen diferentes tecnologías para el reciclaje de las aguas grises: tratamientos físico-químicos (coagulación-floculación, filtraciones), tratamientos biológicos (lodos activos, SBR) o una combinación de los dos (MBR).

Generalmente, estos tratamientos se completan con filtros y sistemas de desinfección. Todos ellos deben asegurar la calidad del agua reciclada en los puntos de uso. Para el diseño del tratamiento se debe determinar la capacidad de captación de aguas grises así como la necesidad de agua reciclada y tener en cuenta el factor limitante [2].

Para instalar un sistema de reciclaje de aguas grises, se debe disponer, indispensablemente, de redes separativas para: las aguas grises, las aguas residuales, las aguas recicladas y las aguas de consumo humano [2].

1.1 Machine Learning

Machine Learning es ideal para:

- Problemas para los que las soluciones existentes requieren muchos ajustes o una larga lista de reglas: un algoritmo de aprendizaje automático a menudo puede simplificar el código y funcionar mejor que el enfoque tradicional.
- Problemas complejos para los que el uso de un enfoque tradicional no ofrece una buena solución.
- Entornos fluctuantes: un sistema de aprendizaje automático puede adaptarse a nuevos datos.
- Obtener información sobre problemas complejos y grandes cantidades de datos [3]

Gracias al avance del área de investigación en Machine Learning, existen enfoques bien establecidos para el aprendizaje automático de máquinas. Cada uno utiliza una estructura algorítmica diferente para optimizar las predicciones basadas en los datos recibidos. Machine Learning es un amplio campo de algoritmos que se agrupan, en general, en tres grandes categorías: aprendizaje supervisado, aprendizaje no supervisado y Reinforcement Learning [4]

El "aprendizaje es supervisado" cuando los datos usados para el entrenamiento incluyen la solución deseada, llamada "etiqueta" (*label*). Algunos de los algoritmos más populares de Machine Learning en esta categoría son la regresión lineal, la regresión logística, *support vector machines*, *decision trees*, *random forest* y redes neuronales.

En Machine Learning label ("etiqueta") se refiere a lo que estamos intentando predecir con un modelo. A una variable de entrada se le llama feature ("característica" o "variable de entrada"). Un modelo define la relación entre *features* y *labels* y tiene dos fases:

-Fase "entrenamiento" ó "aprendizaje". Es la fase de "aprendizaje" del modelo, proporcionando el conjunto de datos de entrada que se etiquetan; de esta manera se consigue que el modelo aprenda iterativamente las relaciones entre las *features* y *labels*.

-Fase de "inferencia" o "predicción". Se refiere al proceso de hacer predicciones mediante la aplicación del modelo ya entrenado a conjuntos de datos no etiquetados.

A continuación se muestra un ejemplo simple de un modelo que expresa una relación lineal entre *features* y *labels* [4]. El modelo puede expresarse de la siguiente forma:

$$y = wx + b \quad Ec.1$$

Dónde:

y es la *label* o etiqueta de un ejemplo de entrada.

x la *feature* de ese ejemplo de entrada.

w es la pendiente de la recta *y* que en general le llamaremos "peso" (*weight*) y es uno de los dos parámetros que debe aprender el modelo durante el proceso de entrenamiento para realizar una inferencia.

b es el punto de intersección de la recta en el eje y llamado "sesgo" (*bias)*. Este es otro de los parámetros que debe ser aprendido por el modelo.

1.2 Red neuronal Multicapa Perceptrón (MLP)

Una red neuronal multicapa Perceptrón (ver figura 2), tiene una capa de entrada (*input layer*), una o más capas compuestas por perceptrones, llamadas capas ocultas (*hidden layers*) y una capa final con varios perceptrones llamada la capa de salida (*output layer*).

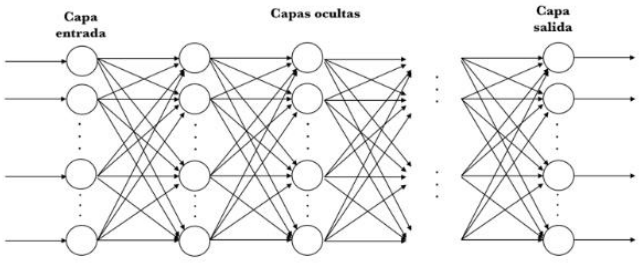


Figura 2. Red neuronal multicapa Perceptrón.

1.3 Red neuronal Perceptrón en Tensorflow-Keras

Clase *Sequential* en Keras

La estructura de datos principal en Keras es la clase *Sequential* [5], que permite la creación de una red neuronal básica. Keras también permite implementar modelos más complejos en forma de grafo que pueden tener múltiples entradas, múltiples salidas, con conexiones arbitrarias en medio.

```
model=tf.keras.Sequential()
```

Para crear el modelo de red neuronal empleando Tensorflow y Keras se debe agregar lo siguiente:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense
```

1.3.1 Configuración del proceso de aprendizaje

A partir del modelo *Sequential*, se definen las capas de manera sencilla con el método *add()*, una vez que se tenga el modelo definido, se procede a configurar cómo será su proceso de aprendizaje con el método *compile( )*.

```
model.add(layers.Dense(64,input_dim=2,activation='sigmoid'))
```

El primero de estos argumentos es la función *loss* usada para evaluar el grado de error entre salidas calculadas y las salidas deseadas de los datos de entrenamiento. Además se especifica un optimizador que permite a la red neuronal calcular los pesos de los parámetros a partir de los datos de entrada y de la función de *loss* definida [6] [7].

Finalmente se debe indicar la métrica para monitorizar el proceso de aprendizaje de la red neuronal.

```
model.compile(optimizer=keras.optimizers.Adamax(0.001),loss='binary_cros
sentropy',metrics=['accuracy'])
```

Una vez definido el modelo y configurado el método de aprendizaje, enseguida debe ser entrenado. Para ello se debe entrenar o "ajustar" el modelo a los datos de entrenamiento que se dispone invocando al método *fit( )* del modelo:

```
model.fit(X, Y, batch_size=100, epochs=2000)
```

1.4 Algoritmo en Raspberry

Para la instrumentación digital se utiliza el sistema embebido Raspberry por su capacidad de procesamiento además de que soporta Python en versiones 2.7 a 3.7 que son las requeridas para la instalación de Tensorflow y Keras.

Una de las desventajas de esta tarjeta es que no posee entradas analógicas mismas que son requeridas por el sensor de pH y turbidez, para solucionar dicho inconveniente se utiliza un módulo convertidor ADS1115 que tiene 4 entradas analógicas con una resolución de 16 bits y cuenta con protocolo I2C que lo hace más robusto.

Para el sistema de adquisición de variables de pH y turbidez se emplean los sensores mostrados en la figura 3. La sonda para medir pH tiene un rango de detección de 0-14 con salida analógica proporcional de 0-4.8V y un tiempo de estabilización de menos de 60 segundos.

Figura 3. Sonda pH y Turbidez

El sensor de turbidez empleado utiliza un elemento óptico para determinar el grado de turbidez en el agua, cuenta con salida analógica de 0-4.5V y una salida digital con umbral ajustable. La turbidez se mide en Unidades Nefelométricas de turbidez, o Nephelometric Turbidity Unit (NTU).

El instrumento usado para su medida es el nefelómetro o turbidímetro, que mide la intensidad de la luz dispersada a 90 grados cuando un rayo de luz pasa a través de una muestra de agua.

1.5 Tratamiento de agua

Separaciones sólido-líquido

Las partículas insolubles, presentes en el agua natural o como resultado de un proceso químico previo, se eliminan por alguno de los siguientes métodos principales: a) sedimentación de las partículas sólidas por diferencia de densidad, b) filtración, donde las partículas sólidas se pueden separar por intercepción mediante una malla de luz apropiada, el uso de un medio sólido poroso o mediante flotación por adición de burbujas de aire que se adhieren a las partículas sólidas y las hacen flotar. La separación de sólidos mediante hidrociclones o centrifugas son sedimentaciones aceleradas sustituyendo la fuerza de gravedad por una fuerza centrífuga auxiliar impuesta sobre la suspensión [8]

1.5.1 Proceso de Coagulación y Floculación

La turbidez y el color son dos características indeseables en las aguas. Ambas suelen estar causados por partículas coloidales. Mientras las partículas en el orden de magnitud de una micra, se pueden considerar en suspensión, y las de una milésima de micra entran en el dominio de moléculas en solución, los tamaños intermedios corresponden al tamaño coloidal. En estos tamaños de partícula las propiedades superficiales y las cargas eléctricas, tienen efectos más importantes que el peso relativo de la partícula en el agua e impiden su simentación.

Las partículas formadas en la coagulación, pueden ser aun pequeñas y de baja densidad. El tamaño de las partículas se puede aumentar con la adición de poli electrólitos, polímeros de moléculas de alto peso molecular y solubles en agua que, por disociación electrolítica en el agua, dan formas iónicas múltiples, capaces de actuar de puentes de unión entre las partículas coaguladas.

Las prácticas de coagulación y floculación (ver figura 4) son tratamientos previos esenciales para muchos sistemas de purificación de agua [8]

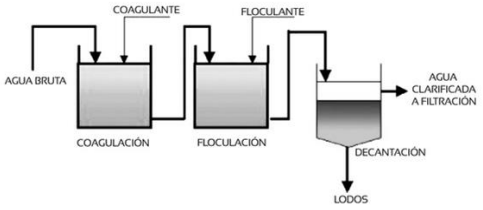


Figura 4. Proceso de coagulación-floculación

En el proceso convencional de coagulación se añade un coagulante al agua fuente para crear una atracción entre las

partículas en suspensión. La mezcla se agita lentamente para inducir la agrupación de partículas entre sí para formar "flóculos". El agua se traslada entonces a un filtro para eliminar y/o minimizar esos flóculos.

1.5.2 Filtración

Consiste en hacer pasar el agua a través de un medio poroso, con el objetivo de retener la mayor cantidad posible de materia en suspensión. El medio poroso tradicionalmente utilizado es un lecho de arena de altura variable dispuesto en distintas capas de diferente tamaño de partícula. Existen varios sistemas de filtración: por gravedad (el agua circula verticalmente por el filtro por simple gravedad) o por presión (el agua se ve forzada a atravesar el filtro mediante presión. En la actualidad estos métodos están siendo desplazados por operaciones con membranas [9].

2. DESARROLLO

2.1 Diagrama general

En la figura 5 se muestra el diagrama general del prototipo, y a continuación se describen las partes principales.

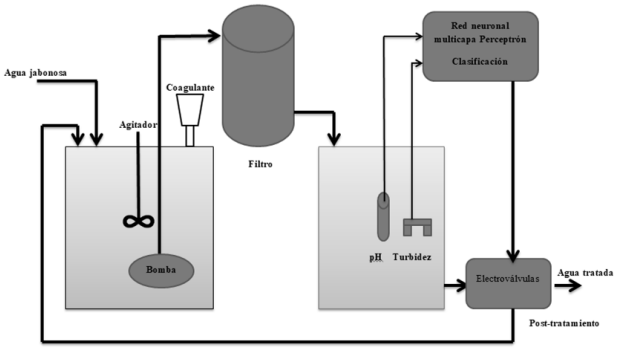


Figura 5. Diagrama general.

2.2 Sistema de coagulación

El sistema coagulante propuesto consta de un agitador y un sistema de inyección de minerales.

Este sistema es el encargado de disolver de manera continua y con una velocidad controlada el coagulante mineral que es inyectado automáticamente por un servomotor. En el diagrama general se aprecia que se conecta una bomba sumergible para una vez que finaliza el proceso coagulante el agua se traslade al sistema de filtrado vertical por gravedad.

2.3 Filtrado

El filtro implementado consta de cinco capas, la primera está conformada por extracto de lenteja, enseguida una capa de carbón activado, después arena y por ultimo una capa de gravilla. Además cuenta con un fondo falso para permitir la salida de agua. Cada capa de material es separada por un medio poroso y la última capa por una placa de plástico perforado.

2.4 Etapa de sensores

Sensor de pH

El sensor de pH proporciona una señal de salida analógica directamente proporcional a la medición del pH a través de un conector BNC.

Cabe destacar que para el uso de este sensor se utilizó un convertidor ADC con protocolo I2C de 4 canales, en el canal A0 se conectó el sensor de pH y en el canal A1 el sensor de turbidez. Para el uso de este ADC se incluyó la librería de Adafruit ADS1X15

Sensor de Turbidez

El sensor de Turbidez es importante para poder determinar la cantidad de solidos suspendidos en el agua (TSS, Total Suspended Solids), parámetro clave para poder determinar la potabilidad del agua, a más turbidez más sólidos en el agua por lo tanto menos potable.

Este sensor mide los niveles de turbidez detectando la proporción de sólidos suspendidos en el agua al medir la transmisión y dispersión de la luz emitida. Concretamente mide a través de un fotodiodo la atenuación de la intensidad de la luz ocasionada por la dispersión debido a sustancias disueltas y no disueltas. El sensor cuenta con salida digital y analógica. En modo digital el umbral de detección es ajustable y en modo analógico el sensor proporciona una salida de voltaje proporcional al nivel de turbidez del agua.

Para realizar los ajustes de este sensor se utilizó agua pura, agua jabonosa, agua jabonosa muy sucia y agua jabonosa con diversos tipos de jabón, esto con el fin de tener parámetros de referencia bien definidos en base a los tipos de agua residual que se va a manipular. En la figura 12 se muestra el montaje de los sensores empleados.

2.5 Proceso de aprendizaje de la red neuronal

Una red neuronal está formada de neuronas conectadas entre ellas; a su vez, cada conexión de la red neuronal está asociada a un peso que dictamina la importancia que tendrá esa relación en la neurona al multiplicarse por el valor de entrada. Cada neurona tiene una función de activación que define la salida de la neurona. La función de activación se usa para introducir la no linealidad en las capacidades de modelado de la red [4].

La parte más relevante en el proceso de aprendizaje es el proceso iterativo de "ir y venir" por las capas de neuronas. El "ir" propagando hacia delante es llamado forward propagation y el "venir" retro propagando información en la red se llama *back propagation*.

La primera fase *forward propagation* se da cuando se expone la red a los datos de entrenamiento y estos cruzan toda la red neuronal para ser calculadas sus predicciones (*labels*).

A continuación se emplea una función *loss* para estimar el error y medir cuán bueno/malo fue el resultado de la predicción en relación con el resultado correcto.

Una vez se tiene la función *loss*, se propaga hacia atrás esta información, de ahí el nombre *back propagation* (retro propagación) Partiendo de la capa de salida, esa información de error se propaga hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. En la figura 6 se resume el proceso de aprendizaje de la red neuronal.

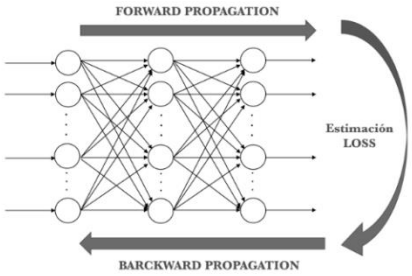


Figura 6. Proceso de aprendizaje de la red neuronal.

2.6 Funciones de activación

Las funciones de activación se utilizan para propagar hacia adelante la salida de una neurona. Esta salida es recibida por las neuronas de la siguiente capa a las que está conectada esa neurona (hasta la capa de salida incluida). La función de activación sirve para introducir la no linealidad en las capacidades de modelado de la red [4].

Sigmoid

La función *sigmoid* permite reducir valores extremos o atípicos en datos válidos sin eliminarlos. Una función sigmoid convierte variables independientes de rango casi infinito en probabilidades simples entre 0 y 1. La mayor parte de su salida estará muy cerca de los extremos de 0.

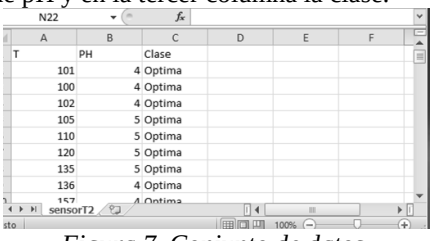
Softmax

La función de activación de *softmax* devuelve la distribución de probabilidad sobre clases de salida mutuamente excluyentes. *Esta función de activación* se encuentra a menudo en la capa de salida de un clasificador.

2.7 Normalización de datos de entrada

Para recabar las muestras de los sensores de pH y Turbidez se emplea el convertidor ADS1115, para las pruebas se utilizaron 105 muestras de ambos sensores y se determinaron tres clases para la clasificación: Optima, Regular, Tratar; la clase "optima" se refiere a que el agua puede ser utilizada para riego, la clase "regular" se refiere a que el agua se puede utilizar para lavado de pisos entre otros y la última clase "tratar" se refiere a que el agua debe volver un segundo ciclo de tratamiento.

Los datos se almacenan en un archivo csv (archivo separado por comas, ver figura 7), en la primer columna se encuentra el sensor de Turbidez (T), en la segunda columna se encuentra los datos de pH y en la tercer columna la clase.



T	PH	Clase
101	4	Optima
100	4	Optima
102	4	Optima
105	5	Optima
110	5	Optima
120	5	Optima
135	5	Optima
136	4	Optima

Figura 7. Conjunto de datos.

Para la lectura del archivo csv se utiliza la herramienta pandas como sigue:

```
datos=pd.read_csv('sensorT2.csv')
df=pd.DataFrame(datos)
X=datos.iloc[:,0:2].values
Y_str=datos.iloc[:,2].values
```

La clase normalmente son caracteres y es debido a esto que se tienen que normalizar los datos.

```
encoder = LabelEncoder()
encoder.fit(Y_str)
Y_num = encoder.transform(Y_str)
Y = to_categorical(Y_num,3)
```

2.8 Creación del modelo de red neuronal

El modelo de red neuronal empleado es un modelo Perceptrón multicapa con dos capas ocultas con 64 neuronas cada una densamente conectadas, la función de activación empleada es sigmoid y softmax en la capa de salida, en la capa uno se debe especificar el número de entradas o características con el comando input_dim, en la capa de salida se debe especificar que se tienen 3 salidas (3 clases).

```
model=tf.keras.Sequential()
model.add(layers.Dense(64,input_dim=2,activation='sigmoid'))
model.add(layers.Dense(64,activation='sigmoid'))
model.add(layers.Dense(3,activation='softmax'))
```

Una vez que se generó el modelo de la red neuronal multicapa se procede a definir los parámetros de optimizador, loss y metrics, a continuación se describe su función.

Loss function. Se emplea para evaluar el grado de error entre las salidas calculadas y las salidas deseadas de los datos de entrenamiento. El objetivo de este parámetro es minimizar esta función para dirigir el modelo en la dirección adecuada.

Optimizer. Permite a la red neuronal calcular los pesos de los parámetros a partir de los datos de entrada y de la función de *loss* definida.

Metrics. Se emplea para monitorear la exactitud del entrenamiento.

Para la compilación del modelo se emplea un optimizador Adam con learnig rate=0.001 y un loss='binary_crossentropy', además se especifican las métricas obtenidas (score %) del modelo.

```
Model.compile(optimizer=keras.optimizers.Adam(0.001),loss='binary_crossentropy',metrics=['accuracy'])
```

Los resultados del modelo se obtienen con el comando model.summary()

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                192
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 195
=================================================================
Total params: 4,547
Trainable params: 4,547
Non-trainable params: 0
```

2.9 Entrenamiento y predicción

Para el entrenamiento de la red neural se emplean la totalidad de las muestras, que en este caso son 105 muestras.

```
model.fit(X,Y,epochs=2000,batch_size=105,verbose=2)
```

```
Epoch 2000/2000
105/105 - 0s - loss: 0.0426 - accuracy: 1.0000
accuracy: 100.00%
```

Para comprobar las clases predichas por el modelo se utiliza el comando:

```
ynn=model.predict_classes(X)
```

Dando como resultado un arreglo que contiene la clasificación (clase) por cada valor de entrada de datos (Turbidez y pH)

```
>>> ynn
array([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, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2], dtype=int64)
```

Una vez que se entrenó la red neuronal y se obtiene un score adecuado (para nuestro caso el 100%) se procede a guardar el modelo de la red neural con el comando

```
model.save("model.h5")
```

2.10 Funcionamiento general de sistema de tratamiento

El proceso inicia al llenar el tanque 1 con agua de enjuague de una lavadora, que al momento se hace de forma manual, una vez llenado se arranca una temporización definida de 3 minutos donde se agita con velocidad alta y a la vez se inyectan minerales (coagulación), después se agita con velocidad media por 3 minutos y se deja reposar sin agitación por 3 minutos más. Enseguida se activa una bomba que trasladada el agua a un filtro vertical (filtrado) que por gravedad el agua se deposita en el segundo tanque, en dicho tanque se tienen los sensores de turbidez y pH que recolectan muestras durante 1 minuto, estas muestra son los atributos de entrada para la red neuronal el cual una vez procesados los datos asigna una de las siguientes etiquetas al agua: “optima”, “regular” o “tratar”. Si es “optima” el agua puede ser usada para riego, si “regular” se puede usar para lavado de pisos u otros usos pero si es “tratar” el agua no se puede reutilizar y requiere otro ciclo de tratamiento.

3. RESULTADOS

Para la etapa de entrenamiento se han realizado diversos modelos y entrenamientos como se muestra en la tabla 2.

Tabla 2. Comparación entre modelos de red neuronal.

Capas ocultas/ No. Neuronas	Función de activación	Optimizador	Loss(función perdida)	epoch	Exactitud(Score) %
3/64	Sigmoid	RMSprop	Binary_crossentropy	2000	94.29
2/64	Sigmoid Softmax	SGD	Binary_crossentropy	1000	66.67
2/64	Sigmoid Softmax	Adam	Binary_crossentropy	2000	100
2/64	Sigmoid	Nadam	Binary_crossentropy	3000	96.19
2/64	Sigmoid Softmax	Adagrad	Binary_crossentropy	4000	66.67
3/64	Sigmoid Softmax	Ftrl	Binary_crossentropy	3000	64
3/64	Sigmoid Softmax	Adamax	Binary_crossentropy	2000	100
3/64	Sigmoid Softmax	Adadelta	Binary_crossentropy	2000	66

En la taba 2 se muestra el comparativo con cada uno de los modelos, siendo el modelo con el optimizador Adam o Adamax los más adecuados para el set de datos y en general para el sistema de clasificación de agua al contar con una exactitud del 100%

4. CONCLUSIONES

Uno de los objetivos primordiales de este prototipo es ahorrar agua en casas habitación empleando el prototipo propuesto.
Se busca innovar en la manera de dar el tratamiento a agua jabonosa empleando para ello redes neuronales para clasificar el uso del agua tratada.
Otra de las ventajas que se busca en el presente prototipo es que sea totalmente automatizado todo el proceso de tratamiento de agua para que el usuario final solo utilice el agua tratada.
Se establecieron parámetros estandarizados para tenerlos en cuenta en los criterios de diseño como son los niveles de coagulante, los requerimientos de filtrado y los sistemas de monitoreo empleando algoritmos inteligentes capaces de predecir los niveles de uso de agua tratada.

5. BIBLIOGRAFÍA

[1] Primera encuesta nacional sobre consumo de energéticos en viviendas particulares (ENCEVI), INEGI, disponible: https://www.inegi.org.mx/programas/encevi/2018/, sitio visitado Marzo 2021
[2]Aguas grises: Origen, composición y tecnologías para su reciclaje, Aqua España, disponible: https://www.aguasresiduales.info/revista/blog/aguas-grises-origen-composicion-y-tecnologias-para-su-reciclaje, sitio visitado Marzo 2021
[3] Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools and Techniques to Build Intelligent Systems, O’Reilly, 2019
[4]Deep learning Introducción práctica con Keras Primera parte, Jordi Torres.AI, disponible:https://torres.ai/deep-learning-inteligencia-artificial-keras/#_ftn2, sitio visitado Marzo 2021
[5] Keras: the Python deep learning API, Keras.io, disponible:https://keras.io/api/models/sequential/#sequential-class, sitio visitado Marzo 2021
[6] Keras: the Python deep learning API, Keras.io, disponible:https://keras.io/api/losses/probabilistic_losses/#binarycrossentropy-class, sitio visitado Marzo 2021
[7] Keras: the Python deep learning API, Keras.io, disponible: https://keras.io/api/optimizers/, sitio visitado Marzo 2021
[8] Miguel Rigola Lapeña, Tratamiento de aguas industriales: Aguas de procesos y residuales, Marcombo, 1989
[9] José Mario Díaz Fernández, Ecuaciones y cálculos para el tratamiento de aguas, Paraninfo, 2018