PuebloCont. Vol. 30(2): 555- 570, 2019
http://doi.org/10.22497/PuebloCont.302.22

ISSN 1991 - 5837 (ed. impresa)
ISSN 2617 - 9474 (ed. en línea)

Espectroscopia infrarrojo y técnicas de machine learning y deep learning para la detección y clasificación de arándanos

Infrared spectroscopy and machine learning and deep learning techniques for the detection and classification of blueberries

Walter Aurelio Lazo-Aguirre[1]

RESUMEN

Las empresas comercializadoras de frutas tienen la necesidad de optimizar la selección y clasificación de las frutas que comercializan, específicamente aquellos productos que se enviarán al extranjero donde se exige altos índices de calidad. Necesitan asegurar que el proceso de selección y clasificación de frutas se realice con precisión para obtener un producto de alta calidad que satisfaga las exigencias de los clientes. Existen equipos que pueden realizar este trabajo de clasificación, pero son muy costosos para su adquisición al igual que su mantenimiento. El presente trabajo tiene por objetivo desarrollar una propuesta basada en la aplicación de espectroscopia con infrarrojo cercano, machine learning y deep learning para la detección y clasificación de frutas para la agroindustria. Específicamente, se toma como caso de estudio, la clasificación de arándanos, con la finalidad de establecer una herramienta alternativa para poder hacer la clasificación de arándanos, que permita reducir el costo, el tiempo y optimizar el proceso de detección y clasificación de frutas. La investigación es de tipo cuasi-experimental, para lo cual se utilizó una muestra de 1000 arándanos. Utilizando un equipo de espectroscopia infrarrojo cercano, NIR, se obtuvo sus espectros electromagnéticos, los cuales fueron digitalizados, se realizó el entrenamiento de una red neuronal utilizando lenguaje de programación python y la plataforma de keras con tensor flow. Luego de realizado el entrenamiento, utilizó la red neuronal para realizar la etapa de testing, con los espectros NIR de nuevas muestras de arándanos, encontrándose que se pueden clasificar los arándanos con una exactitud del 92%.

Palabras clave: Machine learning, deep learning, deep neural networks, espectroscopia, NIR, FTIR, reconocimiento de frutas, arándanos, red neuronal

1 Doctor en Educación - Universidad Privada Antenor Orrego.

ABSTRACT

Fruit marketing companies have the need to optimize the selection and classification of the fruits they sell, specifically those products that will be sent abroad where high quality indices are required. They need to ensure that the selection and classification process of fruits is carried out with precision to obtain a high quality product that meets the demands of the customers. There are machines that can perform this sorting work, but they are very expensive to purchase as well as their maintenance. The purpose of this work is to develop a proposal based on the application of near infrared spectroscopy, Machine Learning and Deep Learning for the detection and classification of fruits for agribusiness. Specifically, the classification of blueberries is taken as a case study, with the purpose of establishing an alternative tool to be able to classify blueberries, which allows to reduce the cost, time and optimize the process of detection and classification of Fruits. The research is of a quasi-experimental type, for which a sample of 1000 blueberries was used, and using a near infrared spectroscopy equipment, NIR, its electromagnetic spectra were obtained, which were digitized, the training of a neural network was performed using Python programming language and the Keras platform with TensorFlow. Then, after the training, he used the neural network to perform the testing stage, with the NIR spectra of new blueberry samples, finding that blueberries can be classified with an accuracy of 92%..

Keywords: Machine learning, Deep learning, Deep neural networks, Spectroscopy, NIR, FTIR, Fruit recognition, blueberries, Neural network

INTRODUCCIÓN

El arándano es una fruta cuyo consumo viene aumentado cada vez más debido su alta capacidad antioxidante y diferentes propiedades beneficiosas para la salud. Los Estados Unidos de América son el principal exportador de arándanos, seguido de Canadá.

En los últimos años, los países del hemisferio sur, han aumentado la exportación de arándanos al hemisferio norte aprovechando las diferencias estacionales en la producción. Pero debido a la larga distancia del envío transoceánico, se requiere que los arándanos frescos, en el país de origen cumplan con tener una alta calidad, para que puedan cumplir con los estándares de calidad a su llegada en el país destino (Leiva, 2013).

La empresa TalSA, es uno de los principales exportadores de arándanos frescos, del Perú. Tiene la necesidad de optimizar la selección y clasificación de las frutas que comercializa, específicamente aquellos productos que se enviarán al extranjero donde se exige altos índices de calidad. Necesita asegurar que el proceso selección y clasificación de arándanos se realice con precisión para obtener un producto de alta calidad que satisfaga las exigencias de los clientes.

Existen equipos que pueden realizar este trabajo de clasificación, pero son muy costosos, sus precios están en el orden de 1'250,000 de euros y darle mantenimiento tiene también un alto costo

Para dar soporte al proceso anteriormente mencionado sobre la producción de arándano, se hace necesario abordar el problema de la detección y clasificación de arándanos frescos en la línea de procesos. Esto se realiza en la etapa de acondicionamiento de arándano en líneas de proceso

En tal sentido, este documento de investigación realiza una propuesta alternativa para la detección y clasificación de arándanos frescos en la línea de procesos.

La clasificación de los arándanos se realiza en forma manual. Desde que se cosecha en el campo hasta el envasado es un proceso manual, en el que la calidad de los frutos depende de la forma como se manipula y la capacidad que tienen los trabajadores para reconocer un fruto que es de calidad y diferenciarlo de un fruto que tiene defectos.

Uno de los principales problemas es que se requiere de un trabajador con experiencia en estas

labores. Asimismo, este trabajador puede realizar sus labores de manera eficiente un determinado tiempo, luego del cual su eficiencia disminuye.

En el presente trabajo, se plantea desarrollar una aplicación utilizando el procesamiento de imágenes por medio de espectroscopia infrarrojo cercano, machine learning y deep learning para la detección y clasificación de arándanos.

El procedimiento que se estudia permitirá identificar las características de los arándanos, de manera no invasiva, a fin de diferenciar frutos en buenas condiciones de aquellos que no lo están, permitiendo separarlos y clasificarlos a fin de seleccionar los productos de buena calidad para su envasado y comercialización.

Arándanos

- Nombre científico: Vaccinium corymbosum L
- Nombre común: Blueberry, mirtilo, blaubeere, anabias, arándano.
- Familia: Ericaceae
- Considerado dentro del grupo de las frutas finas o berries, con alto contenido de antioxidantes y apreciada por sus propiedades nutracéuticas.

El arándano es una baya originaria de América del Norte, donde crece en forma silvestre.

Actualmente es la cuarta fruta más exportada del mundo y su comportamiento es creciente para los próximos cinco años.

Beneficios para la salud

- Poderoso antioxidante.
- Ayuda a prevenir enfermedades como el cáncer y las enfermedades cardiovasculares.
- Tiene poder antiadherente, que hace que las bacterias dañinas para nuestro organismo no se adhieran a las paredes del aparato gastrointestinal.
- Contiene vitamina P, utilizada en problemas de circulación y en afecciones vasculares del ojo.
- Contiene una sustancia conocidas como antocianinas que ayudan a fortalecer el colágeno.
- Contiene flavonoides que potencian la memoria y mejoran el aprendizaje y otras funciones cognitivas.

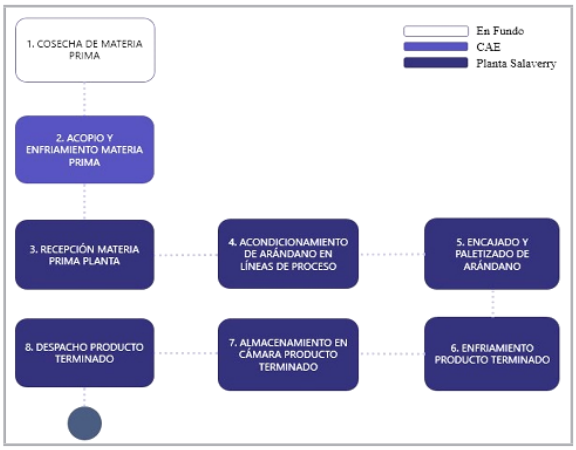
Datos de exportación de arándano en el Perú 2016

- Perú, es el quinto exportador mundial de arándanos
- El Perú tiene a junio 2017 unas 3,800 hectáreas plantadas de arándanos.
- La expectativa de exportación para el 2017 es de 40 mil toneladas contra 29 mil de la campaña anterior.
- Perú tiene el mayor porcentaje de crecimiento anual en áreas de arándanos de América del Sur.
- En los próximo 5 años Perú será el principal exportador de arándanos frescos del Hemisferio Sur
- La región con más superficie cosechada en el Perú es La Libertad
- Región con mayor producción en toneladas, La Libertad (figura 1 y 2).
- Mayor rendimiento por hectárea: La Libertad.
- Perú exporta 70% a EE.UU.

Figura 1: Principales empresas exportadoras de arándanos en La Libertad 2016

Fuente: Empresa TalSA

Figura 2: Proceso arándano fresco



Fuente: Empresa TalSA

Espectroscopia infrarrojo cercano

Para enfocar la formación de la base de datos de las imágenes proponemos el uso de métodos de detección y procesamiento de imágenes del arándano utilizando espectroscopia infrarrojo cercano. La región espectral del infrarrojo cercano (NIR) se extiende desde el extremo de las longitudes más altas del visible (alrededor de 780 ηm) hasta los 3000 ηm (13 000 cm^{-1} hasta 3300 cm^{-1}). Las bandas de absorción en esta zona son sobre tonos o combinaciones de las bandas vibracionales de tensión que se producen en la región de 3000 a 1700 cm. La instrumentación utilizada en la región del IR (infrarrojo) cercano es semejante a la que se emplea para la espectroscopia de absorción ultravioleta/visible. Como fuentes se utilizan las lámparas de tungsteno, y por lo general, las celdas son de cuarzo o sílice fundida como las que se utilizan en el intervalo de 200 a 770 µm.

Machine learning

El aprendizaje automático (Shalev-Shwartz, 2014) es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información suministrada en forma de ejemplos.

Dependiendo del tipo de salida que se produzca y de cómo se aborde el tratamiento de los ejemplos, los diferentes algoritmos de aprendizaje automático se pueden agrupar en el aprendizaje supervisado, que pueden aplicar lo que se ha aprendido en el pasado a nuevos datos, prediciendo el valor de una etiqueta de salida, conocidos otros atributos de entrada. A partir de datos cuya etiqueta se conoce se induce una relación entre dicha etiqueta y otra serie de atributos. Esas relaciones sirven para realizar la predicción en datos cuya etiqueta es desconocida. Esta forma de trabajar se desarrolla en dos fases.

1. Entrenamiento (construcción de un modelo usando un subconjunto de datos con etiqueta conocida)
2. Prueba (prueba del modelo sobre el resto de los datos).

Deep learning

Una red neuronal es un algoritmo que imita el funcionamiento de las neuronas y de las conexiones que hay entre ellas y son entrenadas para que tengan la capacidad de desempeñar una tarea. Se dice que una red neuronal aprende mediante el entrenamiento porque no hay una programación explícita para realizar una tarea, sino que la red se programa sola a partir de ejemplos. Las redes neuronales son el mayor exponente del llamado machine learning o aprendizaje automático. (Precup, D., 2017).

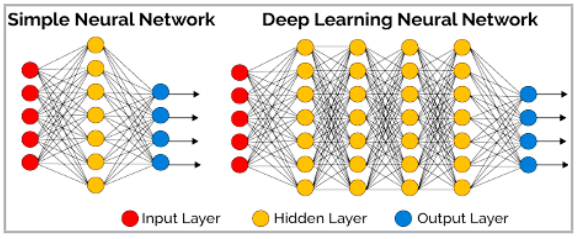
Las redes neuronales pueden aprender a clasificar y a imitar el comportamiento de sistemas complejos. Si se quiere que aprenda a diferenciar entre manzanas y naranjas solo se le tiene que proporcionar unos cuantos ejemplares de ambas frutas e indicarle, a la vez, si se trata de una manzana o de una naranja. Una vez entrenada la red neuronal sabrá si está ante una manzana o una naranja. Lo sabrá aunque las manzanas y naranjas no sean las que se le enseñaron durante el entrenamiento ya que las redes neuronales no memorizan, sino que generalizan. Esa es la clave del aprendizaje de las máquinas.

El estudio de las redes neuronales ha evolucionado hasta lo que hoy se llama deep learning.

Deep learning es una serie de algoritmos emparentados con las redes neuronales que tienen la misma finalidad y un rendimiento mayor que otras formas de machine learning. La mayor diferencia es la capacidad de abstracción (figura 3).

Los algoritmos deep learning son capaces de realizar una abstracción semejante por sí mismos, sin necesidad de que alguien la diseñe previamente. Por esta razón se dice que el deep learning no sólo es capaz de aprender, sino que, además, puede encontrar significado

Figura 3: Los modelos de redes neuronales en deep learning



Fuente: https://www.ackstorm.com/deep-learning-clasificacion-imagenes/

Red neuronal

Es una forma de computación inspirada en modelos biológicos, que hace uso de modelos matemáticos compuestos por un gran número de elementos procesales organizados en niveles.

Se pueden representar como un sistema de computación compuesto por un gran número de elementos simples, elementos de procesos muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas.

Ventajas de las redes neuronales

Basados en su conformación y a sus fundamentos, presentan muchas características semejantes a las del cerebro. En este sentido, son capaces de aprender de la experiencia, de generalizar casos anteriores, de abstraer características esenciales de información que puede ser irrelevante, etc. Esto le origina muchas ventajas y que esta tecnología se esté aplicando en múltiples áreas. Entre estas ventajas se tienen:

Aprendizaje adaptativo.

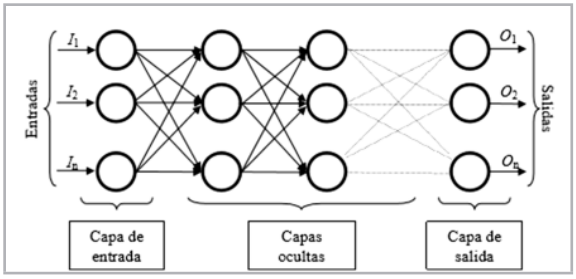
Es la capacidad de poder aprender a efectuar ciertas tareas en base a un entrenamiento o en base a una experiencia inicial. Esto significa que aprenden a realizar determinadas tareas cuando son sometidas a un proceso de entrenamiento utilizando un conjunto de casos que sirven como ejemplos.

Las redes neuronales al ser sometidas a entrenamiento aprenden a reconocer y diferenciar patrones (figura 4). Por esto es que no se requiere de modelos iniciales y tampoco es necesario especificar funciones de distribución de probabilidad.

En el proceso de aprendizaje, se ajustan los pesos de sus enlaces, para que obtengan determinados resultados. Una red neuronal no necesita un algoritmo para resolver un problema, porque puede generar su propia distribución de pesos de sus enlaces al momento del aprendizaje.

Elementos básicos que componen una red neuronal

Figura 4: Ejemplo de una red neuronal totalmente conectada



Fuente: (Soriano et.al , 2010)

En la figura se tiene una red neuronal, constituida por neuronas interconectadas y arregladas en tres capas. Los datos ingresan por medio de la capa de entrada, pasan a través de la capa oculta y salen por la capa de salida. La capa oculta puede estar constituida por varias capas.

Función de entrada

La neurona trata muchos valores de entrada como si fueran uno solo; esto se denomina entrada global. El uso de una función de entrada, la cual se calcula a partir del vector entrada, permite combinar las entradas (in_{i1}, in_{i2}, ...) dentro de lo que sería la entrada global, gin_i (Figura 5)

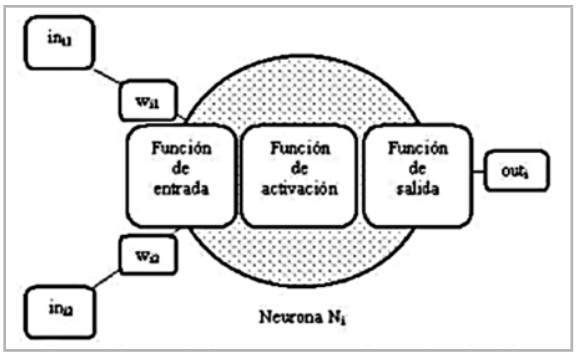
Así, la función de entrada puede describirse en los siguientes términos:

$input_i = (in_{i1} \bullet w_{i1})^* (in_{i2} \bullet w_{i2})^* ... (in_{in} \bullet w_{in})$

donde: * representa al operador apropiado (máximo, sumatoria, producto, etc.), n representa el número de entradas a la neurona N_i , y w_i representa el respectivo peso.

Cada uno de los valores de entrada se multiplican por los pesos asignados a la neurona. Los pesos cambian la medida de la influencia que tienen cada uno de estos valores de entrada. Esto significa que, un valor de entrada grande puede tener una pequeña influencia, si los pesos son los suficientemente pequeños.

Figura 5: Ejemplo de una neurona con 2 entradas y 1 salida.



Fuente: (Soriano et.al , 2010)

La terminología utilizada se explica a continuación:

in_{i1} = Entrada N° 1 para neurona N_i

w_{i1} = peso de ini_1

in_{i2} = Entrada N° 2 para neurona N_i

w_{i2} = peso de ini_2

out_i = salida de neurona N_i.

Vector de entrada = reunión de todas las entradas ini = (in_{i1}, in_{i2}, ..., in_{in}).

Entre las funciones de entrada más utilizadas y conocidas se tiene:

1) Sumatoria de entradas: suma de todas las entradas a la neurona, multiplicadas por sus respectivos pesos.

$$\sum_j \left(n_{ij}\, w_{ij}\right), \quad \text{con } j = 1, 2, ..., n$$

2) Producto de entradas: producto de todas las entradas a la neurona, multiplicados por sus pesos respectivos.

$$\prod_j \left(n_{ij}\, w_{ij}\right), \quad \text{con } j = 1, 2, ..., n$$

3) Máximo de entradas: se considera el valor de entrada más fuerte, multiplicado por su respectivo peso.

$$\underset{j}{\text{Max}}\left(n_{ij}\, w_{ij}\right) \quad \text{con } j = 1, 2, ..., n$$

Función de activación

La neurona biológica tiene un "estado de activación", pudiendo estar activa (excitada) o inactiva (no excitada). De manera análoga, las neuronas artificiales tienen diferentes estados de activación; algunas solamente dos, pero otras pueden tomar cualquier valor dentro de un grupo establecido.

La función activación calcula el estado de actividad de una neurona; transformando la entrada global en un valor (estado) de activación, en el rango de (0 a 1) o de (–1 a 1). Una neurona puede estar inactiva (0 o –1) o activa (1). La función activación, es una función de la entrada global (gini) menos el umbral (Θi).

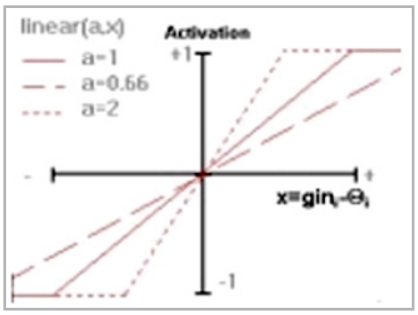
Entre las funciones de activación más comunes, se tiene:

1) Función lineal (figura 6)

$$f(x) = \begin{cases} -1 & x \le -1/a \\ a * x & -1/a < x < 1/a \\ 1 & x \ge 1/a \end{cases}$$

Donde: = $gin_i - \Theta_i$, y a > 0.

Figura 6: Función lineal



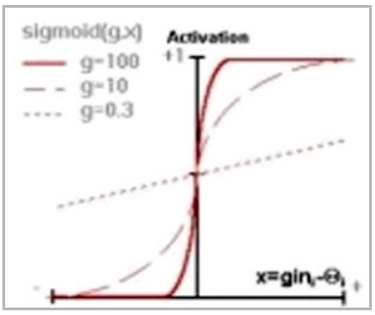
Fuente: (Andrade, 2013)

Los valores de salida con esta función de activación serán: $a \cdot (gin_i - \Theta_i)$, cuando $(gin_i - \Theta_i)$ se encuentra en el rango (-1/a, 1/a). Fuera de este rango la salida será 1 o –1, respectivamente. Si a = 1 entonces se tiene que salida = entrada.

2) Función sigmoidea (figura 7)

$$f(x) = \frac{1}{1 + e^{-gx}}, \text{ con } x = gin_i - \Theta_i.$$

Figura 7: Función sigmoidea



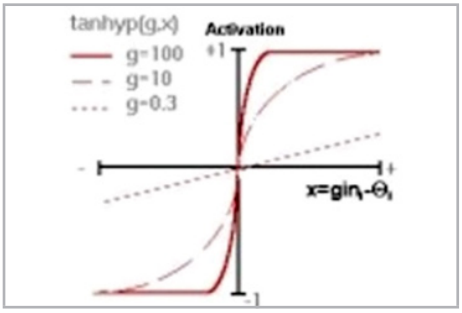
Fuente: (Andrade 2013)

Los valores que produce esta función se encuentran en el rango de 0 a 1. Si se modifica g, se afecta la pendiente de la función.

3) Función tangente hiperbólica (figura 8)

$$f(x) = \frac{e^{gx} - e^{-gx}}{e^{gx} + e^{-gx}}, \text{ con } x = gin_i - \Theta_i.$$

Figura 8: Función tangente hiperbólica



Fuente: (Andrade 2013)

Las salidas de la función tangente hiperbólica están en el rango de -1 a 1. Si se modifica g, se afecta la pendiente de la función.

Estas funciones de activación se utilizan porque se requiere tener una elevada potencia cuando se comienza a trabajar. Cuando se llega a otra capa, nuevamente se requiere incrementar la potencia.

Función de salida

Es el componente final de una neurona. El valor que se obtiene, es la salida de la neurona i (outi); de esta forma, esta función establece el valor que se transfiere a las otras neuronas. Si el valor obtenido es menor que un umbral establecido, no se produce, ninguna salida para la neurona siguiente. No se permite cualquier valor como entrada para una neurona, por consiguiente los valores de salida están en el rango [0, 1] o [-1, 1]. Podrían ser valores binarios {0, 1} o {-1, 1}.

Las funciones de salida más conocidas son:

Ninguna: Denominada función identidad. Su característica es: Salida = entrada

$$\text{Binaria: } \begin{cases} 1 & \text{si act}_i \geq \xi_i \\ 0 & \text{de lo contrario} \end{cases} \text{, donde } \xi_i \text{ es el umbral.}$$

Niveles o capas de una red neuronal

Las neuronas se distribuyen en niveles o capas, con un número determinado en cada una. Considerando esto, se tienen tres tipos de capas:

- Capa de entrada: recibe directamente la información que viene de fuera de la red.
- Capas ocultas: Son capas internas de la red. No interactúan con el entorno exterior. Pueden ser entre cero y un número grande. Las neuronas de estas capas se pueden interconectar de formas distintas, originándose las topologías de redes neuronales.
- Capas de salidas: Son aquellas que se encargan de transferir la información desde la red hacia el exterior.

Una red es totalmente conectada si todas las salidas desde un nivel llegan a todos y cada uno de los nodos del nivel siguiente.

Mecanismos de aprendizaje

Los datos de entrada se procesan a través de la red neuronal para lograr una salida. Las redes neuronales pueden hacer generalizaciones a partir de un conjunto de ejemplos.

Una red neuronal aprende a calcular la salida correcta para cada conjunto de entrada utilizando el conjunto de ejemplos. A esta etapa de aprendizaje se denomina: entrenamiento o acondicionamiento.

El conjunto de datos de ejemplos se denomina conjunto de datos de entrenamiento.

Si durante el aprendizaje no cambian ni la topología de la red ni las funciones, pero los pesos de cada enlace si pueden cambiar; el aprendizaje se denomina adaptación de los pesos. En el aprendizaje la red neuronal modifica sus pesos como respuesta a los datos de entrada.

Los cambios que se pueden dar son la destrucción, modificación y creación de enlaces entre neuronas. La creación implica que su peso pasa a un valor diferente de cero. La destrucción ocurre cuando el peso pasa a ser cero.

Los pesos de los enlaces se van modificando durante el aprendizaje. Dicho proceso termina, es decir la red ya ha aprendido, cuando los pesos se mantienen estables.

Es importante saber cómo se modifican los pesos, cuáles son los criterios para que cambie el valor de un enlace, si se quiere que la red aprenda nueva información. Se distinguen dos métodos de aprendizaje:

a- Aprendizaje supervisado.

b- Aprendizaje no supervisado.

Se debe contar con datos para el entrenamiento y datos para realizar la prueba o test, para la fase de operación o funcionamiento. Una vez que termina la etapa de entrenamiento de la red, los pesos permanecerán fijos o estables.

Aprendizaje supervisado

Su característica es que el proceso de aprendizaje se lleva a cabo con un entrenamiento controlado por un agente externo o supervisor, que es quien establece cual es la respuesta que la red debe producir conociendo los datos de entrada. Si la salida de la red no es la esperada, se debe modificar los pesos de los enlaces para que la salida se acerque a lo esperado (Figura 9).

Figura 9: Influencia de la salida de la neurona Nj en la entrada de la neurona Ni

Fuente: (Soriano et.al , 2010)

Aprendizaje no supervisado

También denominado aprendizaje autosupervisado, ajusta los pesos de los enlaces entre las neuronas, sin intervención de factores externos. La red no recibe información del entorno que le diga si la salida obtenida es correcta o no.

Estas redes, se encargan por si solas, de determinar las características, regularidades, correlaciones o categorías que se puedan establecer entre los datos de entrada. La estructura de las redes y el algoritmo de aprendizaje utilizado, permiten hacer la interpretación de la salida obtenida.

La salida obtenida establece el grado de familiaridad o similitud entre la información de entrada y los resultados.

Detención del proceso de aprendizaje

El entrenamiento termina cuando el error ha alcanzado un mínimo o cuando el error obtenido está por debajo de un determinado umbral, para cada ejemplo dado.

En general, las herramientas de las redes neuronales muestran estos errores por medio de gráficos; los mismos que sirven solamente para tener una idea desarrollo del proceso.

Otra condición para detener el aprendizaje puede ser cuando hayan sido completados cierto número de ciclos y/o pasos de entrenamiento.

Al alcanzar la condición de término del entrenamiento, los pesos permanecerán invariables. Se ha logrado transformar los datos de entrada a los de salida.

Validación de la red neuronal

Luego de concluir el entrenamiento, lo que sigue es comprobar que la red neuronal pueda resolver nuevos problemas, del tipo para los que ha sido entrenada. Para esto se requiere de otro conjunto de datos, llamado conjunto de validación o testeo.

Cada ejemplo del conjunto de evaluación tiene los datos de entrada, conociéndose su solución. La solución calculada para cada ejemplo de validación se compara con la solución conocida, para determinar si la salida de la red neuronal es correcta.

Aplicación de las técnicas de espectroscopia NIR y madurez de frutas

Según Lafuente R.(2015), se puede aplicar las técnicas de espectroscopia de infrarrojo cercano y análisis de Imágenes multiespectrales, para utilizarlas como herramientas que permitan determinar parámetros de calidad de las frutas. Para esto se debe realizar una selección de variables con el fin de proponer modelos de calibración para ser aplicados en las líneas de clasificación de frutas de las empresas del rubro.

El grado de madurez de un fruto se puede determinar en base al contenido de antocianinas a partir de las longitudes de onda que refleja el fruto al ser sometido a radiaciones en el rango del infrarrojo cercano (NIR).

Se comprueba la utilidad potencial de la espectroscopia NIR para predecir el nivel de antocianinas. (RedAgrícola, 2018)

Tecnología de espectroscopia infrarrojo cercano NIR

La Tecnología de Infrarrojo cercano, está basada en una combinación de espectros y métodos matemáticos. La luz NIR es reflejada sobre la muestra y se modifica sutilmente de acuerdo a la muestra.

La modificación espectral es convertida en información mostrando la composición de la muestra.

Clasificación de frutas utilizando deep neural network

Para clasificar una fruta determinada con una red neuronal es necesario extraer características que definan las frutas. Estas características pueden ser el color, la forma, el tamaño, etc. Representar las frutas mediante estas características es una forma de abstracción que debe ser diseñada para que se entrene la red neuronal.

Keras

Keras es una API de redes neuronales de alto nivel, de código abierto, escrita en python y capaz de ejecutarse sobre tensor flow, microsoft cognitive toolkit o theano. Está especialmente diseñada para posibilitar la experimentación en corto poco tiempo con redes de aprendizaje profundo.

Fue desarrollado con un enfoque en permitir la experimentación rápida. Permite pasar de la idea al resultado con el menor retraso posible, facilitando el proceso de investigación.

Keras proporciona una biblioteca de aprendizaje profundo que permite la creación de prototipos de forma fácil y rápida (a través de la facilidad de uso, la modularidad y la extensibilidad). Además, admite redes neuronales convolucionales y redes neuronales recurrentes, así como combinaciones de las dos.

Keras es compatible con varios motores de back-end y no se encierra en un solo ecosistema

Los modelos keras se pueden desarrollar con una gama de diferentes backends de aprendizaje profundo. Es importante destacar que cualquier modelo keras que solo aproveche las capas incorporadas será portátil en todos estos backends: puede entrenar un modelo con un backend y cargarlo con otro (por ejemplo, para la implementación). Los backends disponibles incluyen:

- El backend tensor flow (de Google)
- El backend CNTK (de Microsoft)
- El backend theano

La API de keras es la interfaz oficial de tensor flow, a través del módulo tf. kerasmódulo. (Keras, s.f.).

Tensor flow

Tensor flow es una plataforma de código abierto para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos que permite impulsar el estado del arte en machine learning y poder construir e implementar fácilmente aplicaciones con tecnología machine learning.

TensorFlow, ofrece una API, que sirve a principiantes y expertos para desarrollar aplicaciones en equipos de escritorio, web, servidores, dispositivos móviles y en la nube. Está disponible para investigación y producción. (Tensor flow, s.f.)

Se basa en un sistema de redes neuronales. Esto significa que puede relacionar varios datos en red simultáneamente, de la misma forma que lo hace el cerebro humano.

MATERIAL Y MÉTODOS

Todo el universo de arándanos en proceso de producción de la Empresa TalSA, en el período correspondiente entre julio del 2018 a marzo del 2019.

Muestra

Se elegirá una muestra de 1000 arándanos aleatoriamente y posteriormente se obtendrá sus espectros NIR para formar la base de datos para el aprendizaje y entrenamiento del sistema.

Para las pruebas de clasificación automática, se tomarán uno o varios grupos experimentales de arándanos.

Método

Se dispuso de dos grupos de arándanos, se evaluó a ambos en la variable dependiente, luego a uno de ellos se aplicó el tratamiento experimental y el otro siguió con las tareas o actividades rutinarias.

Hipótesis de trabajo

H1: El uso de espectroscopia con infrarrojo y técnicas de machine learning y deep learning permitirán realizar la Detección y Clasificación de arándanos de forma automática.

- **Variable independiente (VI)**

  El uso de espectroscopia con infrarrojo y técnicas de machine learning y deep learning

- **Variable dependiente (VD)**

  Detección y clasificación de arándanos en forma automática.

Procedimiento

Se explica la metodología a seguir, conformada por las siguientes fases:

1.- Formación de la base de datos

Captura de imágenes espectrales de arándanos, utilizando equipo NIR, digitalización y formación de la base de datos.

2.- Elaboración de la propuesta

Entrenamiento del Sistema de aprendizaje/ clasificación de arándanos, utilizando diferentes algoritmos. Se selecciona el más eficiente.

3.-Aplicación de la propuesta

Pruebas de la clasificación automática sobre los grupos experimental de arándanos.

RESULTADOS Y DISCUSIÓN

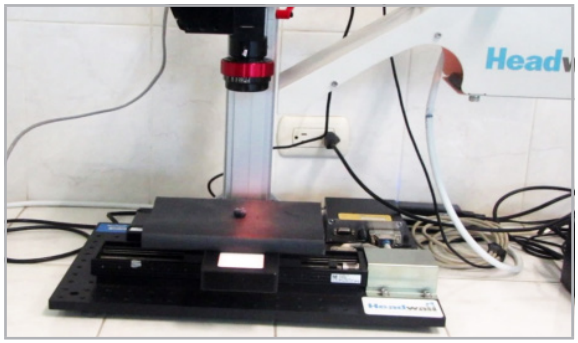
Formación de la base de datos

En primer lugar, se procedió a realizar la captura de imágenes espectrales de los arándanos, utilizando un equipo de espectroscopia de infrarrojo cercano (NIR), este equipo emite una radiación sobre el fruto, obteniéndose las longitudes de onda correspondientes al espectro infrarrojo cercano (figura 10). El equipo de espectroscopia, está conectado a una computadora, la cual tiene instalado el software que viene con el equipo NIR, el mismo que permite que los espectros sean digitalizados automáticamente, formándose la base de datos de longitudes de onda digitalizadas, correspondiente a cada arándano (figuras 11, 12 y 13).

Figura 10: Equipo de espectroscopia NIR y la computadora en funcionamiento

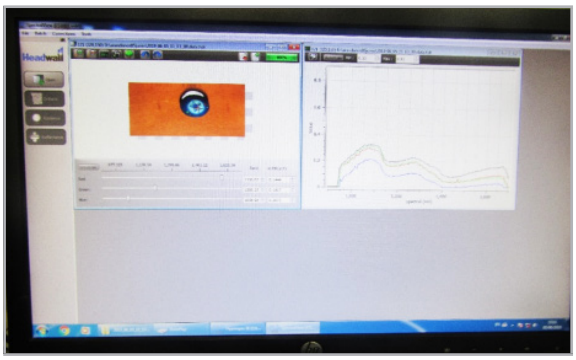
Fuente: Elaboración propia

Figura 11: Proceso de emisión de las radiaciones y captura de espectros NIR



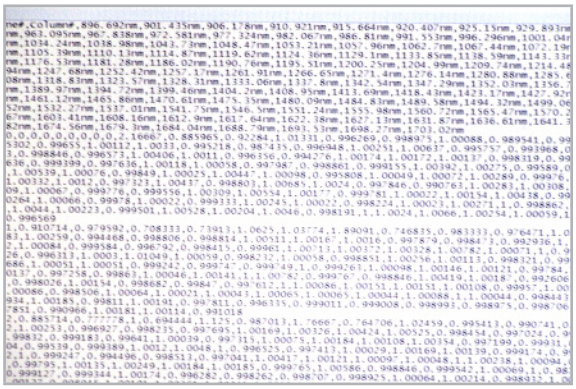
Fuente: Elaboración propia

Figura 12: Proceso de captura de espectros NIR y digitalización



Fuente: Elaboración propia

Figura 13: Datos de espectros digitalizados, correspondiente a un arándano

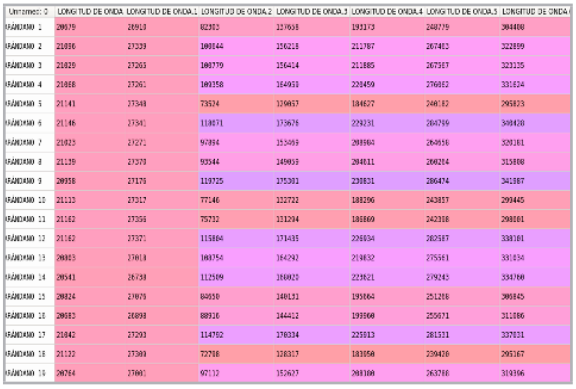


Fuente: Elaboración propia

Elaboración de la propuesta

Para la elaboración de la propuesta, se ha utilizado lenguaje de programación python y las bibliotecas implementadas y disponibles en keras con tensor flow, y lo que se hace es el entrenamiento del Sistema de aprendizaje/clasificación de arándanos. Para esto se tienen los datos obtenidos de los arándanos, digitalizados y tabulados (figura 14):

Figura 14: Datos obtenidos de espectros NIR digitalizados



Unnamed: 0	LONGITUD DE ONDA	LONGITUD DE ONDA.1	LONGITUD DE ONDA.2	LONGITUD DE ONDA.3	LONGITUD DE ONDA.4	LONGITUD DE ONDA.5	LONGITUD DE ONDA.6
ARÁNDANO 1	20079	20918	82303	137658	193173	240779	304408
ARÁNDANO 2	21016	27339	100144	156218	211787	267413	322899
ARÁNDANO 3	21029	27265	100379	156414	211885	267587	323225
ARÁNDANO 4	21068	27261	100358	156493	220459	276882	331824
ARÁNDANO 5	21141	27348	73574	129857	184827	240182	295823
ARÁNDANO 6	21146	27341	110871	173676	229231	284749	340428
ARÁNDANO 7	21023	27271	97894	153469	208864	264656	320101
ARÁNDANO 8	21139	27370	93544	149039	204131	260264	315898
ARÁNDANO 9	20958	27176	119725	175301	230831	286474	341987
ARÁNDANO 10	21132	27317	77146	132722	188296	243837	299445
ARÁNDANO 11	21102	27356	75732	131294	186869	242308	298001
ARÁNDANO 12	21102	27371	115804	171435	226834	282587	338101
ARÁNDANO 13	20883	27018	100754	164292	214832	275561	331034
ARÁNDANO 14	20541	26730	112569	166020	223021	279245	334760
ARÁNDANO 15	20824	27079	84650	140121	195864	251208	306845
ARÁNDANO 16	20683	26898	88916	144412	199860	255671	311086
ARÁNDANO 17	21042	27293	114702	170334	225813	281331	337031
ARÁNDANO 18	21122	27380	72708	128317	183950	239420	295167
ARÁNDANO 19	20764	27001	92112	152627	208188	263788	319396

Fuente: Elaboración propia

En la imagen anterior, las columnas son el conjunto de valores que corresponden a las longitudes de onda y se representaron con la variable X. Adicionalmente se tiene el valor de la última columna, que se identifica con la variable Y, que almacena 1's o 0's. El valor 1 hace referencia a un arándano que se encuentra en óptimas condiciones de color, textura, forma y sabor, mientras que el valor 0 se refiere a un arándano que no cumple con las condiciones antes indicadas, ya sea por demasiada acidez o mal estado del arándano.

En la matriz se tienen las filas y columnas separadas por ',' . Se utiliza ':' para hacer referencia a todas las filas, y en la lista después de la coma se hace referencia a todas las columnas del set de datos, en el caso de la variable dependiente Y, se seleccionan todas las filas y con el valor -1, se hace referencia solamente a la ultima columna.

En resumen, lo que hace este código es guardar el valor de todas las filas de las columnas especificas en las variables 'X' y 'Y'.

```
X = dataset.iloc[:, [2, 19, 37, 38, 55, 56, 73, 74, 91, 92, 109, 110, 111, 127, 128, 145, 146]].values
y = dataset.iloc[:, -1].values
```

Importando las librerías necesarias

En primer lugar, se debe importar las librerías necesarias. Entre estas se tienen las librerías de numpy, tensor flow (este será el framework sobre el que correrá keras), keras y unas librerías necesarias scikit learn, pandas, etc.

Pasos para crear la red neuronal con python

1.- Importar las librerías que serán necesarias para la obtención de los datos

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

2.-Se importa el dataset, utilizando la librería pandas. Para esto se crear una variable y se almacena en ella los datos que se procesarán con la red neuronal

```
dataset = pd.read_csv('arandanosxls2.csv')
```

3.-El siguiente paso es dividir el conjunto de datos o dataset, en un conjunto de variables independientes (X) y la variable dependiente o variable a predecir (Y).

```
X = dataset.iloc[:, [3, 20, 38, 39, 56, 57, 74, 75, 92, 93, 110, 111, 112, 128, 129, 146, 147]].values
y = dataset.iloc[:, -1].values
```

La variable independiente, como se indicó anteriormente, toma los valores 2, 19, 37, etc., y hace referencia a las columnas o longitudes de onda que se seleccionaron del dataset, como las más importantes. Por Ejemplo, aquí se puede ver que la columna correspondiente a 'LONGITUD DE ONDA' y 'LONGITUD DE ONDA 1' que corresponden a las columnas 1 y 2, sus valores no varían o varían muy poco, esto nos indica que estas longitudes de onda no interactúan de manera apreciable con el arándano, así que lo que se hace en el código, es solo considerar aquellas columnas o longitudes de onda que tienen una interacción apreciable con el arándano como se ve en la columna de nombre 'LONGITUD DE ONDA 2' que corresponde al índice 3 del dataset, así que solo se seleccionan las columnas o longitudes de onda que se puede ver que interactúan de manera significativa con el arándano, las cuales serían: 3, 20, 38, 39, 56, 57, 74, 75, 92, 93, 110, 111, 112, 128, 129, 146, 147 y se obtienen los valores para cada arándano.

La variable dependiente Y, es el valor que se desea predecir. Tomará los valores 0 o 1. Se trata de encontrar una relación entre la longitud de onda absorbida por los arándanos y sus características en óptimas condiciones.

4.- Se importan las librerías que servirán para dividir el set de datos en entrenamiento y testing. En este caso se elige un conjunto de test del 25%.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

Se utiliza la librería train_test_split, a la que se le pasa como parámetros el conjunto de datos X y Y. Asimismo se indica que el conjunto de datos para hacer el testing será el 25% del total de datos.

5.-Se importan las librerías para escalar las variables. Esto se hace para evitar que algunos datos dominen sobre otros.

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

6.-Se importan las librerías para crear la red neuronal. Para esto se utiliza keras, que corre sobre tensor flow.

```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

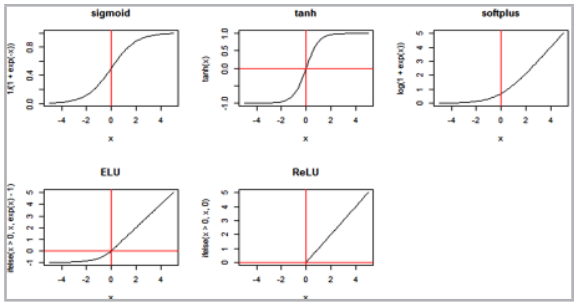
7.- Hay dos maneras de iniciar la red neuronal. Definiendo la secuencia de capas o definiendo el grafo de la relación de capas. En este caso, como va a ser una red neuronal que clasifique arándanos se le pone por nombre classifier. Se elige una secuencia de capas, donde cada capa se define como:

```
classifier = Sequential()
```

8.- Luego se empieza a añadir las capas. La capa de entrada y la primera capa oculta. Con la función add se declara la primera capa. El parámetro input_dim, indica el número de nodos de la capa de entrada. El valor es 171 porque se tienen 171 valores de longitudes de onda por cada arándano, desde 0 hasta 170.

Primero se debe considerar las funciones de activación de cada capa. Las funciones de activación son funciones matemáticas que permitirán a la neurona la activación o la no activación de la misma. Si los datos que llegan a la neurona son suficientes para activar a la neurona, entonces esta despierta, procesa la información que le llega y la envía a la siguiente capa ya sea oculta o final. Las funciones de activación son las que deciden si la información que llega a la neurona es lo suficientemente importante como para despertar a la neurona. Estas son algunas funciones de activación (figura 15):

Figura 14: Algunas funciones de activación



Fuente: https://www.mql5.com/es/articles/3473

Aquí se define la capa de entrada y la primera capa oculta, donde los datos de entrada serán los tipos de datos que se tienen en el set de datos, ya sea nombre, edad, en este caso conjunto de rangos de longitud de onda. Se define el numero de neuronas, esto es a criterio de cada persona, se puede elegir el que permita obtener el mejor resultado; es experimentar en base a prueba y error.

```
classifier.add(Dense(units = 86, kernel_initializer = 'uniform', activation = 'tanh', input_dim = 171))
```

La función de activación de la neurona es una tangente hiperbólica porque ayuda a centrar los datos y ayuda que el aprendizaje para la siguiente capa sea más fácil. El parámetro kernel_ initializer, permite inicializar la red neuronal asignándoles pesos de manera uniforme y luego los va actualizando iterativamente a mejores valores.

El parámetro units, determina el número de neuronas de la primera capa oculta. En este caso es 86, ya que se ha calculado sumando los nodos de la capa de entrada más los nodos de la capa de salida y se divide entre 2. (171+1)/ 2 = 172/2 = 86.

No hay una regla estricta que indique cuantas neuronas se deben colocar.

Luego se define la capa de salida, que tendrá una sola neurona y se utilizará una función sigmoidea porque es útil para una clasificación binaria como 0 o 1, verdadero o falso, arándano en óptimas condiciones o no. Dado que el valor de una función sigmoidea está entre 0 y 1, se puede establecer que el resultado será 1 si es mayor o igual que 0.5, en caso contrario será 0. Por lo tanto, esta función sigmoidea nos dirá que tan probable es que el arándano se encuentre en buenas condiciones.

Se puede definir una segunda capa oculta con el mismo número de neuronas, se puede seleccionar más o menos neuronas, eso queda a elección, asignándole una función específica.

```
classifier.add(Dense(units = 72, kernel_initializer = 'uniform', activation = 'tanh'))
```

Por último, se define la capa de salida, la cual cuenta con una única neurona, ya que según el peso de los datos que le lleguen se activará o no. En este caso se utiliza una función de activación de tipo sigmoidea. Esta función transforma los valores que la activan a una escala de 0, 1, es parecida a la función de tangente hiperbólica, se elige esta y no la función de tangente hiperbólica porque de acuerdo al juicio de expertos es la función que mejor rendimiento tiene en la última capa de una red neuronal de clasificación.

```
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
```

9.-Ahora se compila la red neuronal, con parámetros para optimizar los pesos, elegir el tipo de error a ajustar, y unas métricas para juzgar el rendimiento del modelo.

Se elige el tipo de optimizador ADAM porque mantiene una tasa de aprendizaje para cada

peso en la red neuronal y se adapta por separado a medida que se desarrolla el aprendizaje, en otras palabras, cada peso de cada sinapsis se evalúa por separado y se va mejorando a medida que la red neuronal aprende. Otros motivos para trabajar con el optimizador ADAM es que es computacionalmente eficiente, es apropiado para objetivos que cambian con el tiempo, apropiado para problemas con gradientes muy ruidosos o dispersos y también que es muy adecuado para problemas con un gran conjunto de datos.

Se elige la perdida de "binary cross entropy", ya que mide el rendimiento de un modelo clasificación binario (1 o 0, Si o No, True o False) cuyo resultado es un valor de probabilidad entre 0 y 1. La pérdida de entropía cruzada aumenta a medida que la probabilidad predicha difiere de la observación real.

Por último, se elige el tipo de métrica que se utiliza para monitorizar a la red neuronal la cual es "accuracy", con la cual se evalúa el porcentaje de aciertos, averiguando donde la red neuronal predice en mayor o menor número la correcta selección, es decir permite ver como está aprendiendo la red neuronal.

```
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

```
Epoch 600/600
170/170 [==============================] - 0s 108us/step
- loss: 0.6155 - acc: 0.6412
```

10.- Ahora se entrena a la red neuronal con el conjunto de entrenamiento previamente creado, donde se corrigen los pesos y se seleccionan el número de iteraciones.

Lo que se indica en "batch_size" es cada cuantas iteraciones la red neuronal debe de ajustar su modelo en base a lo aprendido, cada vez que llegue al número de iteraciones requeridas, la red neuronal hará una propagación hacia atrás, esta es la parte donde la capa de salida envía el error hacia atrás a cada neurona que contribuyó a la estimación errónea; pero cada neurona no recibe todo el error, sino que cada neurona recibe una porción del error en base a su contribución o peso en la misma y así cada neurona ajustará los pesos los cuales serán usados en las siguientes iteraciones hasta que se vuelva a producir la propagación hacia atrás.

"Epochs" indica el numero total de iteraciones que hará la red neuronal con los mismos datos, no hay una regla para elegir estos parámetros, esto depende de cada persona, pero se debe tener en cuenta que se podría caer en un sobre ajustes del modelo (lo que indicaría más una memorización que aprendizaje de parte de la red neuronal) si se coloca demasiadas iteraciones o muy poco tamaño de lotes para reajustar el modelo, se debe de buscar el equilibrio.

```
classifier.fit(X_train, y_train, batch_size = 15, epochs = 600)
```

```
Epoch 584/600
170/170 [==============================] - 0s 109us/step
- loss: 0.6226 - acc: 0.6471
Epoch 585/600
170/170 [==============================] - 0s 106us/step
- loss: 0.6220 - acc: 0.6529
Epoch 586/600
170/170 [==============================] - 0s 109us/step
- loss: 0.6234 - acc: 0.6529
Epoch 587/600
170/170 [==============================] - 0s 110us/step
- loss: 0.6259 - acc: 0.6471
Epoch 588/600
170/170 [==============================] - 0s 153us/step
- loss: 0.6224 - acc: 0.6529
Epoch 589/600
170/170 [==============================] - 0s 109us/step
- loss: 0.6256 - acc: 0.6529
Epoch 590/600
170/170 [==============================] - 0s 105us/step
- loss: 0.6230 - acc: 0.6529
Epoch 591/600
170/170 [==============================] - 0s 106us/step
- loss: 0.6225 - acc: 0.6471
```

```
classifier.fit(X_train, y_train, batch_size = 20, epochs = 500)
```

```
loss: 0.1662 - acc: 0.9600
Epoch 338/500
75/75 [==============================] - 0s 67us/step -
loss: 0.1647 - acc: 0.9733
Epoch 339/500
75/75 [==============================] - 0s 85us/step -
loss: 0.1635 - acc: 0.9467
Epoch 340/500
75/75 [==============================] - 0s 70us/step -
loss: 0.1606 - acc: 0.9600
Epoch 341/500
75/75 [==============================] - 0s 68us/step -
loss: 0.1586 - acc: 0.9733
Epoch 342/500
75/75 [==============================] - 0s 93us/step -
loss: 0.1581 - acc: 0.9600
Epoch 343/500
75/75 [==============================] - 0s 68us/step -
loss: 0.1641 - acc: 0.9733
Epoch 344/500
75/75 [==============================] - 0s 63us/step -
```

Se puede ver como va aumentando la exactitud a medida que la red neuronal va aprendiendo. Por ejemplo, se aprecia una exactitud de 96, 97, 94, 96, 97, 96, 97 %.

Uso de la aplicación propuesta en la detección y clasificación de arándanos

Esto se hace en la etapa de testing, utilizando la función predict. Se aplica la red neuronal al conjunto de testing para predecir el resultado y compararlo con los resultados reales y calcular el rendimiento de la red neuronal.

Se elige el umbral de 0.5 porque se considera que es suficientemente probable que el arándano tenga buenas condiciones por encima de 50% de probabilidad, lo cual clasificará al valor de la

predicción como 1 si esta por encima del umbral y como 0 si esta por debajo.

```
y_pred = classifier.predict(X_test)
y_pred = y_pred > 0.5
```

```
loss: 0.0880 - acc: 1.0000
Epoch 496/500
75/75 [==============================] - 0s 69us/step -
loss: 0.0888 - acc: 1.0000
Epoch 497/500
75/75 [==============================] - 0s 82us/step -
loss: 0.0920 - acc: 1.0000
Epoch 498/500
75/75 [==============================] - 0s 64us/step -
loss: 0.0899 - acc: 1.0000
Epoch 499/500
75/75 [==============================] - 0s 101us/step -
loss: 0.0876 - acc: 1.0000
Epoch 500/500
75/75 [==============================] - 0s 269us/step -
```

Se puede apreciar que la exactitud tiende a ser 1 o su equivalente a 100% en la mayoría de los casos.

Análisis estadístico e interpretación de los resultados

Se utiliza una matriz de confusión para ayudar a ver el número de aciertos y fallos y poder predecir el rendimiento de la red neuronal.

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

Aquí se puede ver claramente los 0's que la red neuronal encontró como 0's son 140, y os 1's que la red neuronal encontró como 1's son 90, las demás cifras son los errores que cometió la red neuronal. Con estos valores se puede obtener el rendimiento de la red neuronal simplemente sumando los aciertos y dividiéndolo entre el total de números del conjunto de testing.

	0	1
0	140	10
1	10	90

Rendimiento = (140 + 90) / 250

Rendimiento obtenido de 92 %

Indicadores:

1. Porcentaje de precisión en la detección de arándanos durante la etapa de entrenamiento
2. Eficacia: porcentaje de arándanos clasificados correctamente

Contrastación de la hipótesis

La prueba de hipótesis del proyecto será empleando el método de pre-prueba, pos-prueba.

X: Aplicación Propuesta

O1i ---------- O2i

Pre-prueba Pos-prueba

O1i, O2i: Selección de frutas sin y con el método propuesto.

X: Aplicación propuesta: Uso de espectroscopia con infrarrojo cercano y deep neural networks en la detección y clasificación de frutas en la empresa TalSA

Indicador 1: Porcentaje de precisión en la detección de arándanos durante la etapa de entrenamiento

Pre-Test (O1): Medición previa de la variable dependiente a ser utilizada.

Post-Test (O2): Corresponde a la nueva medición de la variable dependiente a ser utilizada.

O1_____O_____O2

O1: Proceso de clasificación de arándanos sin utilizar la aplicación propuesta.

O: Aplicación la aplicación propuesta

O2: Proceso de clasificación de arándanos utilizando la aplicación propuesta.

En base a los datos recolectados, durante la etapa de entrenamiento se obtuvo una exactitud promedio de hasta 97%.

```
loss: 0.1662 - acc: 0.9600
Epoch 338/500
75/75 [==============================] - 0s 67us/step -
loss: 0.1647 - acc: 0.9733
Epoch 339/500
75/75 [==============================] - 0s 85us/step -
loss: 0.1635 - acc: 0.9467
Epoch 340/500
75/75 [==============================] - 0s 70us/step -
loss: 0.1606 - acc: 0.9600
Epoch 341/500
75/75 [==============================] - 0s 68us/step -
loss: 0.1586 - acc: 0.9733
Epoch 342/500
75/75 [==============================] - 0s 93us/step -
loss: 0.1581 - acc: 0.9600
Epoch 343/500
75/75 [==============================] - 0s 68us/step -
loss: 0.1641 - acc: 0.9733
Epoch 344/500
```

Indicador 2: Eficacia: porcentaje de arándanos clasificados correctamente

Este valor se obtiene al momento de utilizar la aplicación propuesta en modo test, obteniéndose un valor del 92% de eficacia.

	0	1
0	140	10
1	10	90

Rendimiento = (140 + 90) / 250

Rendimiento obtenido de 92 %

De acuerdo a la información proporcionada por los expertos de la empresa, se tiene los siguientes datos

La eficiencia en la selección de los arándanos, se maneja en base a un porcentaje máximo aceptable de defectos que se divide en dos tipos de defectos: calidad y condición. Estos porcentajes varían según un rango de calificativo interno que va desde excelente hasta malo.

Para poder calcular los porcentajes se realiza un muestreo por cada paleta de producto terminado.

Producto terminado: Se retiran entre 1 a 2 Kg (depende del formato que se procese) de distintos niveles de la paleta.

Para que un producto sea considerado bueno, el % de productos con defectos debe ser ≤10%.

Se utilizó la prueba χ^2 de Pearson, que se considera una prueba no paramétrica que mide la discrepancia entre una distribución observada y otra teórica, indicando en qué medida las diferencias existentes entre ambas, de haberlas, se deben al azar en el contraste de hipótesis. La fórmula que da el estadístico es la siguiente:

$$\chi^2 = \sum_i \frac{(\text{observada}_i - \text{teorica}_i)^2}{\text{teorica}_i}$$

Cuanto mayor sea el valor de X^2, menos verosímil es que la hipótesis nula (que asume la igualdad entre ambas distribuciones) sea correcta. De la misma forma, cuanto más se aproxima a cero el valor de chi-cuadrado, más ajustadas están ambas distribuciones.

Se considera N-1, grados de libertad:

Criterio de decisión:

No se rechaza Xo cuando. $X^2 < X_t^2((r-1)(k-1))$, En caso contrario sí se rechaza.

Reemplazando datos: $X^2 = \frac{(92-90)^2}{90} = 0.044$

$X^2 = 0.044 > X_t^2((1-1)(1-1)) = 0$

Como $X^2 = 0.044 > 0$ se rechaza la hipótesis nula y se acepta la hipótesis alterna.

H1: El uso de espectroscopia con infrarrojo y técnicas de machine learning y deep learning permitirán realizar la detección y clasificación de arándanos de forma automática en la empresa TalSA.

CONCLUSIONES

- Se propuso una aplicación utilizando detección y procesamiento de imágenes por medio de espectroscopia con infrarrojo cercano y las deep neural networks para la detección y clasificación de arándanos en la empresa TalSA.

  Para esto, se utilizó lenguaje de programación python y las bibliotecas que brindan keras y tensor flow, que son plataformas de código abierto para el aprendizaje automático. Cuentan con un ambiente integral y flexible de herramientas, bibliotecas y recursos que permite impulsar el desarrollo de aplicaciones en machine learning y poder construir e implementar fácilmente aplicaciones con tecnología machine learning.
- Como parte del proceso se obtuvo los espectros NIR de las muestras de arándanos, se digitalizaron y se formó la base de datos que sirvió para entrenar las redes neuronales.
- Se utilizó la aplicación propuesta en la detección y clasificación de arándanos.
- Se realizó el análisis estadístico de los datos obtenidos encontrándose que la aplicación propuesta permite detectar y clasificar los arándanos con una exactitud de 92%. Lo que nos indica que efectivamente si se puede utilizar la espectroscopia NIR y machine learning y deep learning para poder clasificar arándanos.

BIBLIOGRAFÍA

1. Aguilera J. (2017). Deep learning para clasificación en imágenes. Recuperado el 28 de octubre del 2018 de: https://www.ackstorm.com/deep-learning-clasificacion-imagenes/
2. Alava Ingenieros. (2018). Sistemas de imagen para el Infrarrojo Cercano (NIR) para diversas aplicaciones. Recuperado el 8 de noviembre del 2018 de: http://www.grupoalava.com/ingenieros/actualidad/sistemas-de-imagen-para-el-infrarrojo-cercano-nir-para-diversas-aplicaciones/
3. Andrade, T. (2013). Estudio de los principales tipos de redes neuronales y las herramientas para su aplicación- Andrade. Ecuador. Universidad Politécnica Salesiana. Recuperado el 10 de abril del 2019 de: https://dspace.ups.edu.ec/bitstream/123456789/4098/1/UPS-CT002584.pdf
4. Chollet, Fran (2015). Keras. La biblioteca de aprendizaje profundo de Python. Recuperado el 9 de marzo del 2019 de https://keras.io/

5. Dolz Z. (2008). Evaluación de la calidad de fruto en manzano: estudio de métodos no destructivos de análisis. Zaragoza. España. Recuperado el 9 de marzo del 2019 de: http://digital.csic.es/bitstream/10261/18601/1/Proyecto%20Pilar%20Dolz.pdf

6. Flores K. , Paz, P., Sánchez M., Pérez D., López M., Contreras M., Mejía J., Guerrero J., Garrido A. (2009). "Determinación no destructiva de parámetro de calidad de frutas y hortalizas mediante espectroscopía de reflectancia en el infrarrojo cercano. Universidad de Córdova. España. Recuperado el 9 de marzo del 2019 de: https://helvia.uco.es/xmlui/bitstream/handle/10396/2070/9788478019427.pdf

7. Gómez, G. (2018). Aprendizaje profundo. México. Inaoe. Recuperado el 10 de abril del 2019 de:https://ccc.inaoep.mx/~pgomez/cursos/IC-I/acetatos/ApProfundo.pdf

8. Keras-Pandas (s.f.). Guía del usuario. Recuperado el 9 de marzo del 2019 de: https://keras-pandas.readthedocs.io/en/latest/intro.html

9. La fuente V. (2015). Aplicación de las técnicas de Espectroscopia Vis/NIR y de imágenes de retrodifusión de luz láser a la evaluación del estado de madurez de melocotón, manzana y cereza. (Tesis de Doctorado en Ciencias Químicas). Universidad de Zaragoza, España. Recuperado el 11 de octubre del 2018 de: http://digital.csic.es/bitstream/10261/126618/1/LafuenteV_TD-EEAD_2015.pdf

10. Nogales J. (2016). Estudio del estado de madurez y la aptitud enológica en uva mediante análisis de imagen hiperespectral. (Tesis de Doctorado en Ingeniería Química). Universidad de Sevilla – España. Recuperado El 11 de noviembre del 2018 de: https://idus.us.es/xmlui/bitstream/handle/11441/55531/Tesis_Julio_Nogales_Bueno_sin_publicaciones.pdf?sequence=1

11. Pastor, J. (2016). 17 expectativas de cómo el machine learning va a cambiar el mundo. Recuperado el 10 de abril del 2019 de: https://www.xataka.com/robotica-e-ia/17-expectativas-de-como-el-machine-learning-va-a-cambiar-el-mundo

12. Precup, D. (2017) Introduction to Machine Learning. McGill University. Montreal. Recuperado el 10 de Julio 2018 de: file:///H:/Tesis/DEEP%20Learning/Precup%20-%20dlss-intro-2017%20Introduction%20to%20machine%20Learning.pdf

13. RedAgrícola. (2018). Una exacta medición de madurez en arándano. Recuperado el 9 de marzo del 2019 de: http://www.redagricola.com/pe/una-exacta-medicion-de-madurez-en-arandano/

14. Salomón, F. (2015). Tecnología NIR, sus Usos y Aplicaciones. V Congreso Argentino de Nutrición Animal - CAENA 2015. Recuperado el 20 de septiembre del 2018 de: https://www.engormix.com/balanceados/articulos/tecnologia-nir-sus-usos-t32534.htm

15. Samanamud Ríos, V. (2001). Estadística aplicada a la educación. Trujillo: Universidad Privada Antenor Orrego.

16. Sladojevic S., Arsenovic, M., Anderla A., Culibrk, D. Stefanović, D. (2016). Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification. Computational Intelligence and Neuroscience. 2016. 1-11. 10.1155/2016/3289801. Serbia. Recuperado el 26 de Mayo del 2018 de http://dx.doi.org/10.1155/2016/3289801

17. Shalev-Shwartz, E., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms (2014). Cambridge University Press, NY. Recuperado el 20 de febrero e 2019 de: https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf

18. Smola A., Vishwanathan, S. (2008). Introduction to Machine Learning. United Kingdom: Cambridge University Press. Recuperado el 9 de abril del 2019 de http://alex.smola.org/drafts/thebook.pdf

19. Soriano, A., Soria, E., Martin, J.(2010).Redes neuronales artificiales. España. Universidad de Valencia. Recuperado el 10 de abril del 2019 de: http://ocw.uv.es/ingenieria-y-arquitectura/1-2/libro_ocw_libro_de_redes.pdf

20. TensorFlow (s.f.). An end-to-end open source machine learning platform.Recuperado el 9 de marzo del 2019 de: https://www.tensorflow.org/overview

wlazoa@upao.edu.pe

Faena
Ángel Quispe Gonzales