e -ISSN: 2661-6688 Volumen 2, Número 2. (Julio - Diciembre 2020)

Revista Técnico - Científica PERSPECTIVAS

Clasificador de Productos Agrícolas para Control de Calidad basado en Machine Learning e Industria 4.0

Agricultural Product Classifier for Quality Control based on Machine Learning and Industry 4.0

[β]Carlos A Guaillazaca G., [α]Valeria K. Hernández A.

[β]Escuela Superior Politécnica de Chimborazo, Riobamba, Ecuador.
[α]Steam-Up Groups, Riobamba, Ecuador
[β]carlos.guaillazaca@espoch.edu.ec, [α]vh_steamup@gmail.com

Resumen- **En la actualidad, las técnicas empíricas en la producción agrícola ecuatoriana para la identificación y clasificación de productos no son suficientes para alcanzar estándares de calidad con normas de inocuidad alimentaria y así lograr cubrir la demanda de un mercado internacional. Este trabajo presenta un sistema capaz de supervisar, identificar y clasificar la calidad de productos del sector agrícola, mediante la aplicación de técnicas de soft computing y algoritmos de machine learning que contribuyen a la identificación de imágenes en tiempo real. La investigación permitió implementar algoritmos de clasificación de K vecinos más cercanos para etiquetar los productos según su calidad y enviar los reportes en tiempo real a una aplicación web mediante el protocolo MQTT. Los productos utilizados para este estudio fueron bananas, naranjas, plátano verde y manzanas. Los resultados obtenidos permitieron determinar el mínimo número de imágenes requeridos para el entrenamiento de los modelos de identificación y las tasas de error de identificación durante la etapa de validación.**

Palabras Clave- **Automatización, Inteligencia Artificial, KNN, MQTT, Tecnología Agrícola.**

Abstract- **Currently, empirical techniques in Ecuadorian agricultural production are not enough to achieve quality changes with food safety standards and thus meet the demand of an international market. This work presents a system capable of supervising, classifying, and controlling the quality of products in the agricultural sector, by applying soft computing techniques and machine learning algorithms that affect the identification of images in real time. The research will implement classification algorithms of nearest K neighbors to label the products according to their quality and send the reports in real time to a web application using the MQTT protocol. The employed products in this study were bananas, oranges, green bananas, and apples. The obtained results allow to determine the minimum number of images required for training the identification models and the identification error rates during the validation stage.**

Keywords- **Automation, Artificial Intelligence, KNN, MQTT, Agricultural Technology**

I. INTRODUCCIÓN

En la actualidad las empresas miden su rendimiento competitivo en base a su capacidad de producción y a los estándares de calidad de sus productos manufacturados, donde predominan en estas variables la tecnología empleada para una producción exitosamente rentable. Las pequeñas y medianas empresas ecuatorianas viven una desventaja muy grande en comparación a la capacidad de producción de empresas multinacionales[1]. En la gran mayoría, sus procesos se caracterizan por ser manuales y usar conocimientos empíricos, la maquinaria y tecnología empleada es casi nula [2], su cadena de producción es lenta y no llega a satisfacer en su totalidad la demanda y requerimientos de su grupo objetivo de clientes[3].

Para una producción exitosa es indispensable proveer a estas empresas vulnerables con tecnología al alcance de sus necesidades y posibilidades económicas [4], ya que el principal problema representa los altos costos de inversión en maquinaria y soluciones de hardware y software [5].

El nivel de automatización en los procesos de producción de las PYMES ecuatorianas es sumamente bajo [6], lo cual genera un gran problema con la calidad de los productos y tiempos de producción no adecuados para satisfacer la demanda interna, lo que la exportación de sus productos se convierte en una utopía inalcanzable.

La investigación e implementación de nuevas tecnologías ha permitido una gran adaptabilidad en diversos sectores, mejorando así notablemente los procesos productivos agroindustriales. El desafío científico actual radica en conocer los métodos tradicionales de agricultura, aprender de las incertidumbres que se enfrentan día a día los agricultores y ejecutar soluciones adaptativas, por lo que ha sido natural tratar de integrar y mejor estas tareas con técnicas basadas en el conocimiento de Inteligencia Artificial [7].

Abordando el libro *Inteligencia Artificial: un enfoque moderno* [8] de los autores Stuart Russell y Peter Norving nos plantean la unificación de su trabajo con relación a los agentes inteligentes en las máquinas. Considerando esto, la

Fecha de Recepción. 03 - 06 - 2020 *Fecha de Aceptación. 08 – 06 – 2020*
DOI: https://doi.org/10.47187/perspectivas.vol2iss2.pp21-28.2020

definición para Inteligencia artificial es "*el estudio de agentes que reciben percepciones del entorno y realizan acciones*" [8].

De esta manera, un agente inteligente debe cumplir ciertas características para denotar cualidades importantes que ayuden a resolver problemas. Para ello, en [9] se plantean las características que debe exhibir un agente inteligente como: Reactivo, Proactivo, Social y Autónomo.

Pero también, a los agentes se les puede atribuir otras características como: continuidad temporal, racionalidad, veracidad y adaptabilidad, de tal forma que les capacite a resolver problemas [10], [11].

Para el estudio de clasificación de patrones en imágenes, la metodología soft-computing es un aliado primordial y punto de partida para el desarrollo de este estudio. El soft-computing tiene un enfoque moderno basado en una verdad parcial, mediante cálculos aproximados y estocásticos [12], [13].

En esta misma línea, las redes neuronales artificiales (RNAs) han sido la herramienta del soft-computing más utilizada para tareas que requieren el reconocimiento de patrones en un conjunto de datos, como imágenes. Una RNA está formada de varios niveles y números de neuronas artificiales, que se constituyen en la unidad de procesamiento, cuyo modelo matemático permite tener varias entradas de datos y una sola saluda que es la ponderación de sus entradas [14], [15]. La conexión de varias neuronas dentro de una RNA constituye una poderosa herramienta de cálculo paralelo, pero capaz de entregar salida aproximativas y no definitivas. Dado que la RNA es comparable a un cerebro biológico [16], esta también debe ser entrenada para que realice una tarea en específico. Es importante mencionar que existen varias estructuras de RNAs, así como varios algoritmos de entrenamiento [17].

Una aplicación bastante difundida de las RNAs es en el denominado algoritmo "k-nearest neighbors" (KNN). El objetivo de este algoritmo es establecer matemáticamente una relación de distancias entre todos los elementos de un conjunto de datos y agrupar en sectores bien definidos todos aquellos elementos que comparten características comunes, lo que se traduce en distancias circundantes más cercanas. De esta manera, para el agente inteligente que implementa dicho algoritmo le es posible identificar a que grupo pertenece cada nuevo elemento en un conjunto [18].

Por otro lado, el denominado "internet de las cosas" (IoT) e industria 4.0 están permitiendo cada vez más la introducción de maquinaria autónoma e inteligente en el sector industrial [19]. En esta área, es imprescindible que las máquinas sean dotadas de una capacidad de comunicación y socialización con otras máquinas, e inclusive entre las partes conformantes de un mismo sistema. En la actualidad, el protocolo MQTT (Message Queuing Telemetry Transport, en inglés) siendo ampliamente usado para la comunicación maquina a máquina (M2M) [20]. Este protocolo se basa en un sistema de subscripciones de servicios, publicaciones y notificaciones de avisos dentro de dichos servicios. Esto favorece a que varias máquinas puedan ser notificadas al mismo tiempo sobre un evento en particular dentro de un servicio al que están subscritas.

Esta investigación se basa principalmente en la utilización de técnicas de técnicas computacionales de inteligencia artificial, como las RNAs y algoritmo KNN, para implementar un agente inteligente sobre un sistema de clasificación de productos agrícolas que permita identificar productos buenos y malos a partir de una entrada de video. El sistema se complementa con un subsistema de acción sobre la línea de producción, a partir de la decisión realizada por el subsistema de inteligencia artificial.

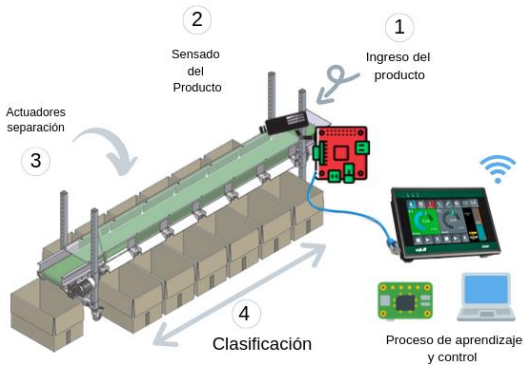

Fig. 1. Arquitectura General del Sistema.

El resto de este documento se organiza de la siguiente manera. En la sección II se presenta la metodología de implementación del sistema de clasificación descrito, enfocándose principalmente en el diseño y entrenamiento de la RNA para ejecutar el algoritmo KNN, además de la plataforma de comunicación MQTT y la elaboración de la estructura hardware de todo el sistema. En la sección III se presentan los principales resultados de la validación y evaluación del clasificador 4.0 obtenidos bajo diferentes escenarios de pruebas. El documento concluye con las conclusiones respectivas de la metodología aplicado y resultados obtenidos.

II. METODOLOGÍA

En esta sección se presenta el proceso metodológico de desarrollo del presente trabajo para diseñar y construir un sistema capaz de supervisar, controlar y clasificar cualquier producto manufacturado, objeto o materia prima procedente del agro.

A. Requisitos generales

Previo al proceso de control y supervisión de calidad, se considera que el prototipo debe ser inducido en una etapa de entrenamiento supervisado para que el sistema reconozca y diferencie los productos malos y buenos de una manera intuitiva. Dicha información debe ser registrada en la nube para que posteriormente pueda ser monitoreada remotamente por agentes encargados de producción y clientes. Los requisitos que se busca reflejar en dicho prototipo de sistema son:

- Capacidad de clasificar productos buenos, regulares y malos.
- Versatilidad para la migración a otros productos.
- Interfaz humano-máquina intuitivo, flexible, sencillo y fácil de interactuar.
- Supervisión de proceso y de los productos clasificados de manera remota.
- Fácil instalación.
- Flexibilidad para la utilización de del sistema mediante

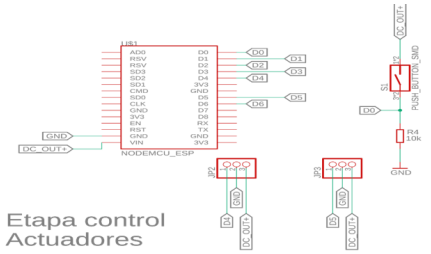

Fig. 4. Etapa disposición para el control de actuadores.

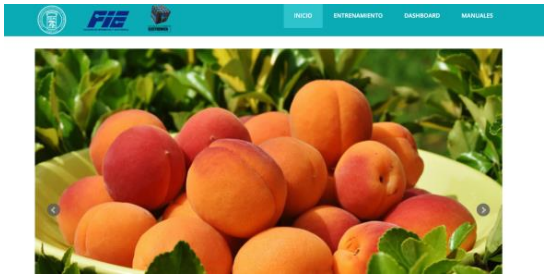

Fig. 5. Aplicación Web.

una aplicación web.

- Mantener una comunicación fiable entre el clasificador y un Dashboard.

El esquema general de la presente aplicación se resume en la figura 1. El proceso de clasificación que ha sido considerado se basa en 3 etapas: Ingreso, Detección y Actuación. Las acciones objetivas del sistema están limitadas al accionamiento de 3 actuadores. Estos se interpretan como las variables de control y, en consecuencia, serán los encargados de clasificar y seleccionar productos buenos, regulares y malos.

B. Selección del Hardware

Como elemento central de procesamiento se selecciona una tarjeta Raspberry Pi 4. Por medio de los periféricos de comunicación embebidos en la tarjeta se implementa vías para el envío y recepción de datos. Esto ayuda a tener un procesamiento de la información directamente en la nube, y todos los datos de control viajarán hasta el broker MQTT, el cual se enlaza a la tarjeta de desarrollo Iot ESP8266. Esta última emite las órdenes a los diferentes actuadores empleados en el proceso. En la figura 2 se muestra el diagrama de conexiones del modulo ESP8266 y los actuadores del sistema.

C. Implementación del software.

Para la implementación del interfaz web que se muestra en la figura 3 fue necesario el desarrollo utilizando los lenguajes de programación HTML, JavaScript y CSS, mediante el entorno de desarrollo (IDE) Átomo. Por otro lado, para la creación del modelo RNA se utilizó Python con jupyter notebook. El desarrollo de la plataforma software fue orientado a lograr un correcto funcionamiento y flexibilidad para el usuario usando el algoritmo del diagrama de la figura 4.

En dicho diagrama se ilustran tres bloques principales: Entrenamiento, Dashboard y Manuales. Estos bloques constituyen las ventanas de navegación de la interfaz web. La ventana “Entrenamiento” contiene el modelo de red neuronal

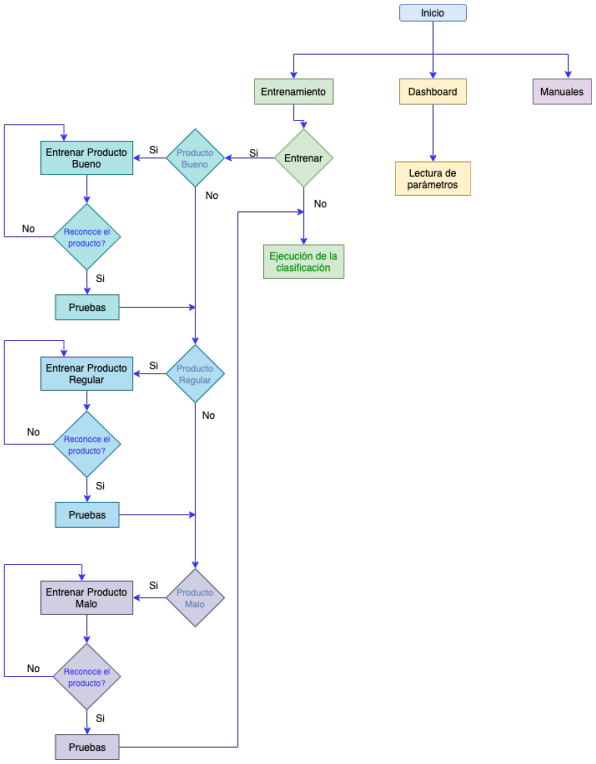

Fig. 2. Diagrama de flujo de la aplicación web.

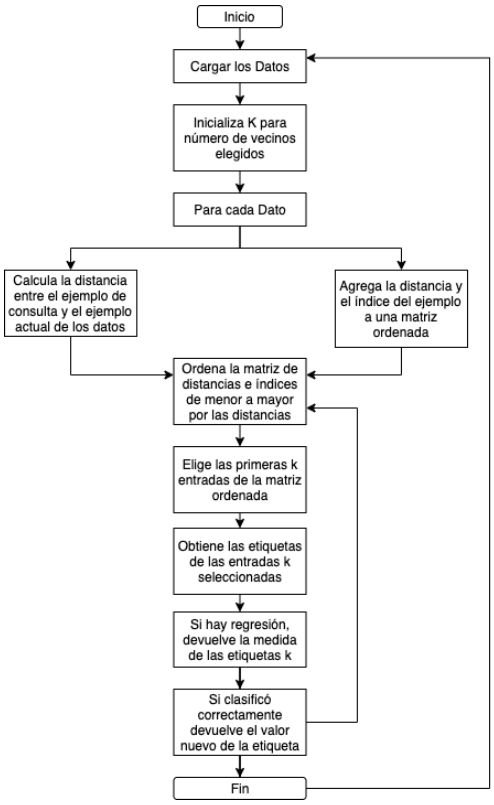

Fig. 3. Diagrama fe flujo, funcionamiento del algoritmo KNN.

y el algoritmo de entrenamiento de la KNN que sigue diagrama de flujo ilustrado en la figura 5. Para esta tarea fue necesario implementar la librería ml5.js que desarrolla algoritmos de aprendizaje automático para entornos web.

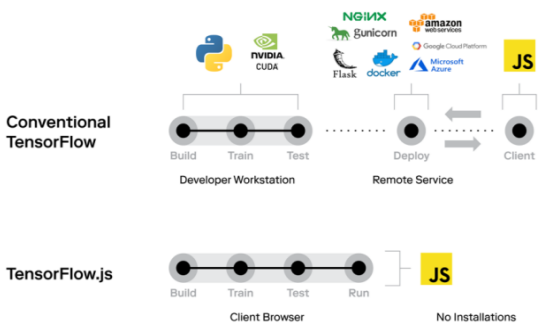

Fig. 6. Metodología para implementar Tensor Flow web.

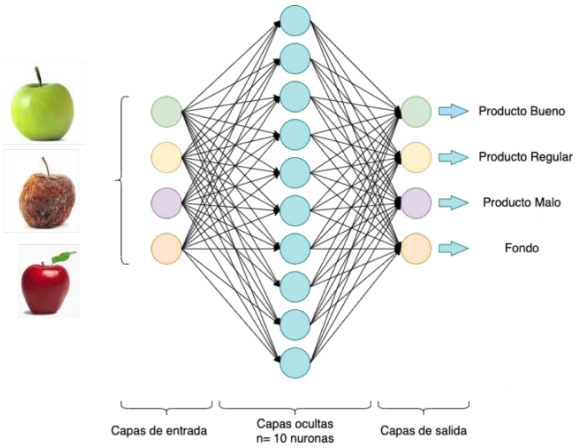

Fig. 7. Representación general de la red neuronal para clasificar imágenes según la calidad del producto.

La flexibilidad y la ventaja de ml5.js es que nos permite programar un modelo en JavaScript como lo hace Keras en Python [21]. La diferencia radica en que Keras necesita un servicio adicional para trasladar el modelo a plataformas Web como se observa en la figura 6, mientras que ml5.js está desarrollado para cubrir esa necesidad sin utilizar más recursos o servicios de terceros.

Una red neuronal convolucional (CNN, por sus siglas en inglés) funciona mediante la extracción de características de las imágenes que ingresan a su estructura a manera de datos. Esto elimina la necesidad de extracción manual de las funciones de activación. La CNN aprende mientras la red entrena con un conjunto de imágenes, que en el caso de este estudio son imágenes de frutas en diferente estado. Esto hace que los modelos de aprendizaje profundo sean más precisos para las tareas de visión por computador. Para esto se utilizó un algoritmo de Tensor Flow y las librerías de Keras como se detalla a continuación:

Paso 1: como primer paso se debe configurar el computador con el ambiente de desarrollo Python, para esto se debe acceder al sitio web www.anaconda.org y descargar anaconda 2020.02 Python 3.7 importante usar esta versión.

Paso 2: para utilizar el estándar actual en la codificación de Python se debe seleccionar la versión 3.7 ya que la 2.7.X se encuentra descontinuada. Para ejecutar algoritmos de machine learning es necesario instalar los siguientes paquetes: Tensorflow, Keras y Numpy. Este proceso se lo realiza desde el terminal.

Paso 3: instalado todo lo necesario, se abre el entorno de desarrollo Spyder que está integrado en la plataforma de Anaconda-Navigator y se inicia declarando lo siguiente:

Sys y **os** son librerías que permiten interactuar con archivos del sistema hasta el editor de código.

ImportDataGenerator se encarga de pre procesar las imágenes que entran al algoritmo.

Optimizers es un optimizador adjunto del algoritmo.

Sequential es una librería que permite crear redes neuronales secuenciales, es decir que cada una de las capas están en orden.

Convolution2D y **MaxPooling2D** son las capas donde se harán las convoluciones.

BackeSignum.nd de keras, este ayuda a que cuando haya una sesión que está corriendo en segundo plano, la elimina y optimiza recursos físicos del computador para empezar el entrenamiento fresco del nuevo algoritmo.

Paso 4: se declara los parámetros para entrenamiento que se explican a continuación:

Épocas, de la línea 20 variable que indica el número de veces que se va a iterar sobre todo el Data Set del entrenamiento.

Altura y **longitud** son variables que van a delimitar el tamaño uniforme de las imágenes.

Batch_size es el número de imágenes que se va a enviar al computador a procesar en cada uno de los pasos.

Pasos, variable que indica el número de veces que se va a procesar la información en cada una de las épocas.

Validation_step, son los pasos de validación que al final de cada una de las épocas se van a ejecutar 200 pasos con el Data Set de validación antes mencionado.

FiltrosConv1 y **FiltrosConv2**, son filtros que se aplica en cada convolución en la primera convolución agrega una profundidad de 32 y en la segunda convolución una profundidad de 64.

Tamano_filtro1 y **tamano_filtro2**, son tamaños de los filtros que se agregan a cada convolución (altura, longitud).

Tamano_pool, este es el tamaño del filtro que se va a utilizar en Maxpooling2D.

Clases, son las salidas de la red neuronal o el conjunto de especies que se va a clasificar.

Lr, este es el learning rade, indican el tamaño de los ajustes de la red neuronal para acercarse a una solución lógica.

Paso 5: se realiza el preprocesamiento de las imágenes de entrada y se unifican las características tal como se muestra en la figura 8 y figura 9, respectivamente. Para eso se utilizan las siguientes herramientas:

Class_names, representa a class_names=['Maduras' , 'Buenas'].

Labels, representa a la concatenación de las etiquetas generadas de cada producto con el código: **Labels =np.concatenate([etiqueta_maduras , etiqueta_buenas).**

Paso 6: en este punto se crea la red neuronal convolucional CNN con lo siguiente:

Cnn, es la variable que genera una red secuencial, es

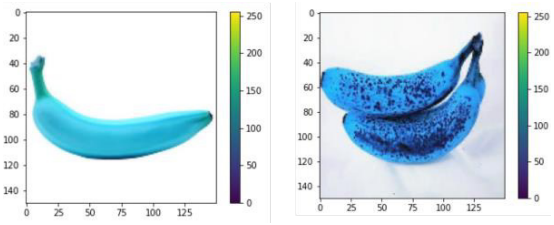

Fig. 10. Resultado de los parámetros anteriores.

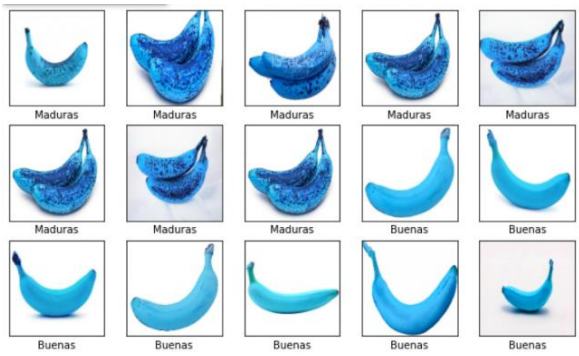

Fig. 11. Detalle del logotipo de la revista Perspectivas.

Fig. 12. Interfaz web para entrenamiento de la CNN.

decir, que son varias capas apiladas entre ellas.

Adicionalmente, se fuerza mediante código para que la primera capa de la CNN haga una convolución con los números de filtros mencionados en el paso 4. El **padding** configura las esquinas, el **input_shape** define las variables necesarias en el paso 4 y por último se tiene una función de activación. Posterior a esto, se agrega una capa de **pooling**.

Es importante considerar que se puede agregar más capas para la CNN para una mejor predicción, pero tiene costos computacionales muy altos. Esta opción dependerá de forma empírica sobre la configuración del hardware del computador utilizado para la etapa del entrenamiento.

Finalmente, para completar el tratamiento de los datos, entrenamiento y aprendizaje de la CNN se consideraron ciertos parámetros de configuración como son:

Flatten ayuda a que las imágenes profundas se conviertan a planas, es decir, que solo se va a tener una sola dimensión.

Dense con 256 neuronas que reciben la información de la línea anterior.

Dropout se declaró el 0.5, esto quiere decir que durante el entrenamiento se apaga el 50% de las neuronas cada paso,

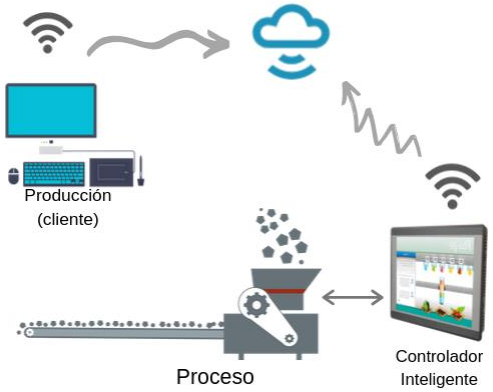

Fig. 8. Esquema de interacción de comunicaciones del sistema.

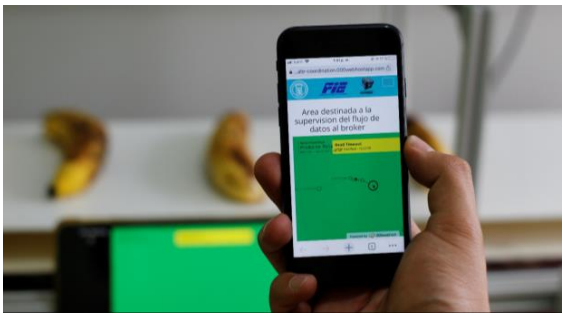

Fig. 9. Supervisión del sistema desde el navegador móvil.

con esto se evita que el algoritmo aprenda un solo camino para una clasificación de un producto en específico y genere alternativas cuando se enfrente a nueva información.

La última capa se declara con una clase de 3 neuronas (variable declarada en el **paso 4**). La herramienta **softmax** al final de la CNN determina el porcentaje de cada una de las salidas, y aquella con el número mayor es la que predomina en la clasificación. En la figura 10 se puede apreciar la interfaz de entrenamiento dentro de la aplicación web desarrollada para este estudio.

D. Enlazar los datos con el broker MQTT

El protocolo MQTT es muy importante al momento de implementar aplicaciones de internet de las cosas (IoT, por sus siglas en ingles). El IoT permite establecer una conexión con varios clientes y servicios de manera instantánea sin mayores recursos mediante un broker. En la figura 11 se presenta el esquema de comunicación entre los diferentes equipos que conforman la presenta aplicación.

Los datos que emite el control de la aplicación web, son enviados mediante el protocolo MQTT hacia la nube. A través de dicho protocolo, un cliente se puede conectar por medio de un enlace inalámbrico con el controlador ESP8266 para tener cuenta del estado de inventario y estado de los productos clasificados tal como se muestra en la figura 12. El bróker MQTT utilizado en este estudio fue "*broker.shiftr.io*" a través del puerto 80.

E. CAD estructura.

La estructura mecánica, previo ser llevada a la

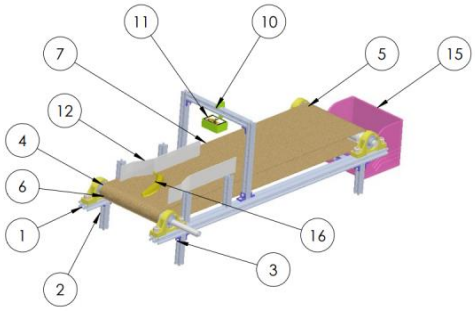

Fig. 13. Diseño CAD estructural del prototipo.

Tabla 1
MATERIALES EMPLEADOS EN EL PROTOTIPO DE CLASIFICADOR

N.º DE ELEMENTO	ELEMENTO	CANTIDAD
1	Vslot 20 x 40 x 1000mm	2
2	Vslot 20 x 20 x 250	8
3	Angulo Simple serie 20	22
4	Chumacera 20mm ¾”	4
5	Eje Teflón diámetro 40mm	1
6	Eje Teflón principal 40mm	1
7	Banda sintética PVC 190mm	1
8	Tornillo 5 x 0.8 x 10	44
9	Tornillo 5 x 0.8 x 5	1
10	Vslot 20 x 20 x 100mm	5
11	Case Cámara	1
12	Guías de acrílico	2
15	Caja producto 1	1
16	Fruta	1

implementación dentro del presente estudio, fue diseñada en su totalidad utilizando software CAD como se muestra en la figura 13. Cada uno de los elementos constitutivos del diseño son numerados y detallados a cabalidad en la tabla 2 para que pueda ser replicado sin dificultad. Además, en la figura 14 se muestra el prototipo implementado físicamente y en fase de pruebas de clasificación.

III. RESULTADOS

En esta sección se presentan los resultados obtenidos durante la etapa de entrenamiento y validación del prototipo de clasificador inteligente. La validación fue llevada a cabo con 4 productos agrícolas como son: bananas, naranjas, plátano verde y manzanas. La selección de estos productos tuvo como finalidad probar el rendimiento del algoritmo KNN con varios escenarios de identificación diferenciada por colores y formas. Cada producto se evaluó de acuerdo con tres variables de identificación (Producto Bueno, Producto Regular y Producto Malo).

A. Etapa de Entrenamiento

Para llevar a cabo el entrenamiento del algoritmo KNN fue necesario contar con un data-set o conjunto de datos conocidos que describan las características de cada producto tal como se mencionó anteriormente. Dado que los conjuntos

Fig. 14. Prototipo implementado de clasificador.

Tabla 2
PRUEBAS EN ETAPA DE ENTRENAMIENTO

PRUEBA	*# DE IMÁGENES*
Prueba 1	5
Prueba 2	10
Prueba 3	15
Prueba 4	20
Prueba 5	25
Prueba 6	30
Prueba 7	35
Prueba 8	40
Prueba 9	45
Prueba 10	50

de imágenes debían ser sumamente específicos, se optó por recolectar una base de datos propia y personalizada a las necesidades del proyecto. Sin embargo, en esta primera etapa no era claro cuantas imágenes serían suficientes para entrenar a las CNNs.

Respecto a esto, se realizaron varias sesiones de entrenamiento con un numero incremental de imágenes para determinar el punto en el cual el error de identificación y clasificación es minimizado. La tabla 2 presenta el número de pruebas que se realizaron, y el respectivo número de imágenes utilizadas, con la finalidad de encontrar un modelo de RNA con el algoritmo KNN óptimo para la etapa de validación. El objetivo de realizar estas pruebas fue identificar el nivel mínimo requerido de entrenamiento del modelo y no saturar al algoritmo con entrenamiento innecesario.

En la figura 15 se muestran las curvas de aprendizaje de las CNN con el algoritmo KNN resultantes de la fase de entrenamiento para los productos: (a) Bananas, (b) Naranjas, (c) Plátano Verde y (d) Manzanas. Las líneas en color anaranjado, gris y amarillo corresponden a los casos de productos de calidad “Buena”, “Regular” y “Mala”, respectivamente. Como se puede apreciar, en la mayor parte de casos probados, el error de identificación durante la fase de entrenamiento alcanzó el valor de 0% a partir de la “Prueba 9”. Una excepción a esto, lo constituye el caso del producto Plátano Verde con calidad “Buena” que mantiene un error del 10.0%, incluso en la “Prueba 10” (ver figura 15.d).

Debido a que, el producto “Plátano Verde” no alcanzó un error de identificación de 0% en todos los casos de calidad de producto probados, se optó por realizar dos pruebas extras con dicho producto. Al final se obtuvo un error de identificación de 0% (para todos los casos de calidad) en la Prueba 11 con 55 imágenes. Es importante mencionar que este procedimiento fue importante para lograr estimar el número mínimo de imágenes requeridas para el entrenamiento del modelo en cuestion. El resumen del

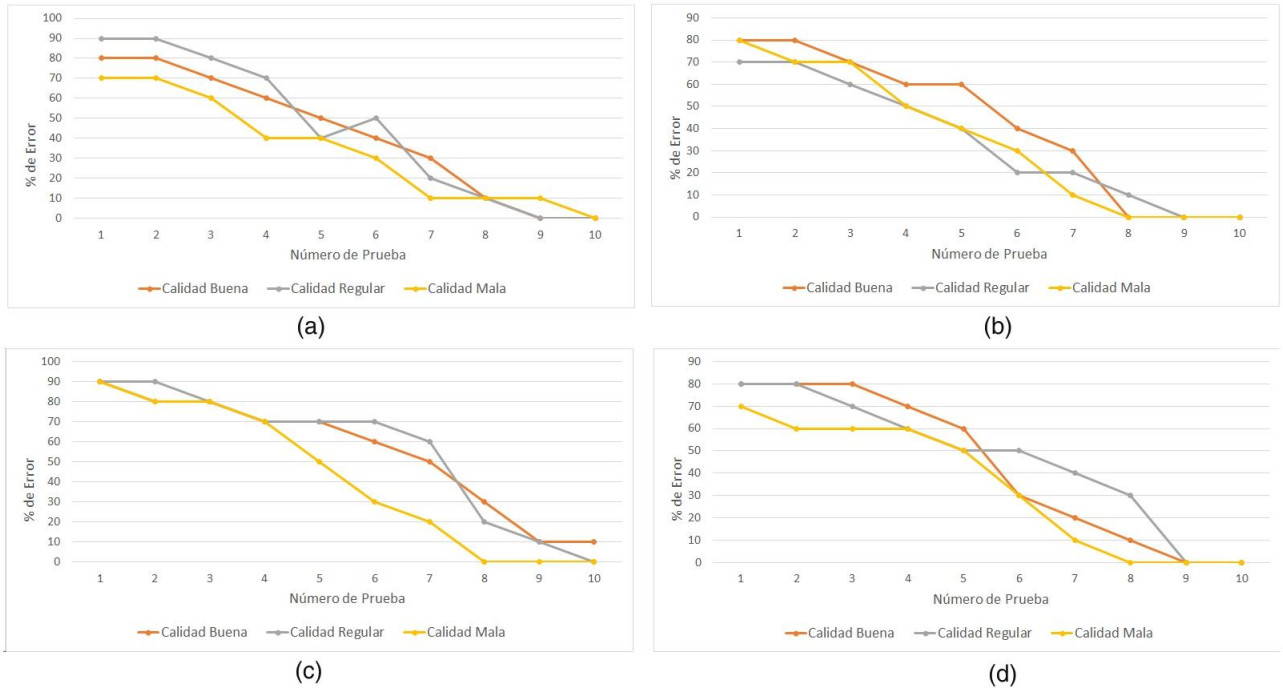

Fig. 15. Curvas de aprendizaje de las CNNs con algoritmo KNN respecto al error porcentual de identificación de los productos: (a) Banano, (b) Naranjas, (c) Plátano Verde y (d) Manzanas.

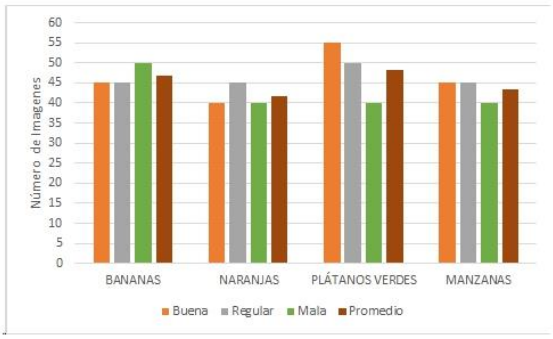

Fig. 16. Resumen del mínimo número de imágenes requeridas para el entrenamiento de las CNN por cada producto.

Tabla 3
RESULTADOS DE PRUEBAS DE VALIDACIÓN DEL CLASIFICADOR INTELIGENTE

No.	Productos							
	Banana		Naranja		Plátano Verde		Manzana	
	No. Error	% Error	No. Error	% Error	No. Error	% Error	No. Error	% Error
1	0	0	1	11.1	1	11.1	0	0
2	0	0	0	0	0	0	0	0
3	0	0	1	11.1	1	11.1	0	0
4	0	0	0	0	1	11.1	0	0
5	0	0	0	0	0	0	0	0
Media		0		4.44		6.67		0

mínimo número de imágenes con el que se obtuvo el menor error de identificación para cada caso probado en este estudio se muestra en la figura 16. Además, se ha incluido el promedio de imágenes requeridas para el entrenamiento de las CNNs por cada producto.

Como se puede apreciar, el producto más facil de identificar por el algoritmo KNN durante el entrenamiento fue "Naranjas" debido a que en promedio requirió un menor número de imágenes con ~41.67. Mientras que el producto que más dificultad presento en su identicación fue "Plátanos Verdes", que en promedio requirió un mayor numero de imágenes con ~48.33. Por otro lado, los modelos de los productos "Bananas" y "Manzanas" requieren para su entrenamiento un mínimo promedio de imágenes de ~46.67% y ~43.33%, respectivamente.

En base a este precedente, y previo la etapa de validación, se entrenaron cuatro nuevos modelos de identificación (uno por cada producto) utilizando el mínimo número de imágenes determinado en la figura 16. Las imágenes fueron aleatoriamente escogidas en cada conjunto de datos, aunque de manera proporcional respecto a la calidad del producto.

B. Etapa de Validación.

En esta etapa, se utilizaron 9 nuevas imágenes por cada producto (3 imágenes por cada categoría de calidad) y se procedió a evaluar en 5 ocasiones los modelos. En cada prueba de validación, el orden de ingreso de las 9 imágenes fue cambiado aleatoriamente. Cabe recalcar que las imágenes utilizadas en esta etapa son nuevas para el modelo de identificación y no fueron utilizadas durante la etapa de entrenamiento.

El resumen de los resultados de validación es presentado en la tabla 3. Como se puede observar, los modelos que presentan errores durante la validación son los encargados de identificar y clasificar la calidad de "Naranjas" y "Plátano Verde" con un promedio de errores de 4.44% y 6.67%, respectivamente. Los errores presentados por el modelo encargado de clasificar la calidad de "Plátano Verde" son consistentes con el hecho que dicho producto fue más difícil

identificarlo durante el entrenamiento. Sin embargo, en el caso del producto "Naranjas", sus errores pueden ser debidos a que, a pesar de que dicho modelo presentó el mejor rendimiento durante la fase de entrenamiento, el número de imágenes es aún bajo y ligeramente insuficiente respecto los otros casos.

IV. Conclusiones

Un prototipo de clasificador de productos agrícolas fue implementado utilizando redes neuronales convolucionales CNN, entrenadas a partir del algoritmo de aprendizaje supervisado "K-Nearest-Neighbor" o KNN. Además, en el desarrollo se emplearon herramientas de código abierto para IoT e industria 4.0 como el protocolo de comunicación M2M denominado MQTT. El objetivo del estudio fue obtener modelos de identificación de la calidad buena, regular y mala de productos agrícolas como bananas, naranjas, plátano verde y manzanas. El entrenamiento de los modelos se llevó a cabo con 10 pruebas y un conjunto de 50 imágenes por cada producto y calidad. Los resultados obtenidos permitieron determinar que durante el entrenamiento el producto mejor identificado fueron las naranjas, requiriendo solamente ~41.67 imágenes para entrenar su respectivo modelo. Mientras que por otro lado, el producto más difícil de identificar fue el plátano verde, que requirió ~48.33 imagenes. Finalmente, los mejores modelos de identificación fueron validados mediante 5 pruebas y un nuevo conjunto de 9 imágenes por cada producto. Los resultados obtenidos determinaron que los modelos de identificación para los productos bananas y manzanas no presentaron errores en la validación, mientras que los modelos de identificación de los productos naranja y plátano verde presentaron errores de 4.44% y 6.67%, respectivamente. A través del estudio se concluye que el algoritmo KNN se ajustó a la identificación de formas y colores, con una adecuada etapa de entrenamiento supervisado, lo cual lo hace flexible para utilizarlo en aplicaciones en tiempo real.

Agradecimientos

Un particular agradecimiento al Ing. Edwin Altamirano como director del Trabajo de Titulación que dio origen al presente trabajo. Además, se agradece de forma especial al Dr. Jorge Hernández Ambato por su tutoría y asesoría en la elaboración del presente manuscrito.

Referencias

[1] «Poca tecnificación reduce competencia en el agro - NOV. 09, 2004 - Economía - Historicos - EL UNIVERSO», nov. 09, 2004.

[2] «Información Agroambiental y Tecnificación Agropecuaria |», 2018. https://www.ecuadorencifras.gob.ec/informacion-agroambiental/ (accedido may 28, 2020).

[3] M. T. T. Livio, «LAS TAXONOMÍAS SOBRE LOS AGENTES ECONÓMICOS EN EL AGRO ECUATORIANO 1965-2015. CONTEXTO, SUPUESTOS TEÓRICOS, APORTES Y LÍMITES.», UCE, Quito, Ecuador, 2018.

[4] «La tecnificación es una de las alternativas a la crisis arrocera», *El Comercio*.

[5] S. Sherwood y M. Paredes, «El futuro como producto del presente: caso de estudio sobre la modernización agrícola en Carchi, Ecuador.» Grupo FARO, 2013, Accedido: may 28, 2020. [En línea]. Disponible en: https://n9.cl/g8g9.

[6] H. Jácome Estrella y K. King, Eds., *Estudios industriales de la micro, pequeña y mediana empresa*, 1. ed. Quito, Ecuador: FLACSO Ecuador, 2013.

[7] «Artificial Intelligence and Environmental Decision Support Systems», p. 15.

[8] S. Russell, P. Norvig, J. M. Corchado Rodríguez, y L. Joyanes Aguilar, *Inteligencia artificial: un enfoque moderno*. Madrid: Pearson Educación, 2011.

[9] M. Wooldridge y N. R. Jennings, «Intelligent agents: theory and practice», *Knowl. Eng. Rev.*, vol. 10, n.º 2, pp. 115-152, jun. 1995, doi: 10.1017/S0269888900008122.

[10] S. Franklin y A. Graesser, «Is It an agent, or just a program?: A taxonomy for autonomous agents», en *Intelligent Agents III Agent Theories, Architectures, and Languages*, Berlin, Heidelberg, 1997, pp. 21-35.

[11] P. H. Winston, «Artificial intelligence: a perspective», 1990, p. 11, [En línea]. Disponible en: https://www.semanticscholar.org/paper/Artificial-intelligence%3A-a-perspective-Winston/75d03d764d147b751de9fb835fb5935aa08e0013.

[12] I. Dzitac, F. G. Filip, y M.-J. Manolescu, «Fuzzy Logic Is Not Fuzzy: World-renowned Computer Scientist Lotfi A. Zadeh», *Int. J. Comput. Commun. Control*, vol. 12, n.º 6, p. 748, dic. 2017, doi: 10.15837/ijccc.2017.6.3111.

[13] J. R. Hagerty, «Lotfi Zadeh Tried to Adapt Math and Computers to a World of Ambiguity», *Wall Street Journal*, sep. 22, 2017.

[14] I. A. Basheer y M. Hajmeer, «Artificial neural networks: fundamentals, computing, design, and application», *J. Microbiol. Methods*, vol. 43, n.º 1, pp. 3-31, dic. 2000, doi: 10.1016/S0167-7012(00)00201-3.

[15] D. Mehta, X. Zhao, E. A. Bernal, y D. J. Wales, «Loss surface of XOR artificial neural networks», *Phys. Rev. E*, vol. 97, n.º 5, p. 052307, may 2018, doi: 10.1103/PhysRevE.97.052307.

[16] M. van Gerven y S. Bohte, «Editorial: Artificial Neural Networks as Models of Neural Information Processing», *Front. Comput. Neurosci.*, vol. 11, 2017, doi: 10.3389/fncom.2017.00114.

[17] R. F. López y J. M. F. Fernández, *Las Redes Neuronales Artificiales*. Netbiblo, 2008.

[18] C. G. Cambronero y I. G. Moreno, «ALGORITMOS DE APRENDIZAJE: KNN & KMEANS», p. 8.

[19] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, y M. Hoffmann, «Industry 4.0», *Bus. Inf. Syst. Eng.*, vol. 6, n.º 4, pp. 239-242, ago. 2014, doi: 10.1007/s12599-014-0334-4.

[20] A. Stanford-Clark y H. L. Truong, «MQTT For Sensor Networks (MQTT-SN) Protocol Specification», p. 28, 1999.

[21] «Keras | TensorFlow Core», *TensorFlow*, 2019. https://www.tensorflow.org/guide/keras?hl=es (accedido may 17, 2020).