ISSN 1870-4069

Detección de pimiento morrón utilizando TinyML

Jesús A. Martínez-Vargas, Said Polanco-Martagón,
Yahir Hernández-Mier, Marco A. Nuño-Maganda

Universidad Politécnica de Victoria,
Sistemas Inteligentes,
México

{1930337, spolancom, yhernandezm, mnunom}@upv.edu.mx

Resumen. En este trabajo se presenta un sistema TinyML y visión artificial para la detección de pimientos morrones. Se utilizó una red neuronal convolucional con arquitectura MobileNet la cual se implementó en un dispositivo de borde de bajo costo y baja potencia, la ESP32CAM. Para el proceso de entrenamiento se optó por utilizar la técnica de transferencia de conocimiento (transfer learning) de pesos pre entrenados con el conjunto de datos COCO 2017 para disminuir el tiempo de cómputo requerido. Se utilizó el framework Edge Impulse con un conjunto de datos con un total de 1910 imágenes de pimientos morrones y otros objetos que pudiesen encontrarse en un huerto. La exactitud del modelo quantizado a int8 fue de 91.12 % contra el conjunto de prueba. Los resultados obteniendos se muestran favorables y alentadores para trabajos futuros.

Palabras clave: Aprendizaje profundo, tinyML, sistemas embebidos, cómputo de borde.

Detecting Bell Peppers through TinyML

Abstract. In this work, a TinyML and computer vision system for the detection of bell peppers is presented. A convolutional neural network with MobileNet architecture was used, which was implemented on a low-cost and low-power edge device, the ESP32CAM. For the training process, it was chosen to use the transfer learning technique of pre-trained weights with the COCO 2017 dataset to reduce required computation time. The Edge Impulse framework was used with a dataset with a total of 1910 images of bell peppers and other objects that could be found in a chili garden. The accuracy of the model quantized to int8 was 91.12% against the test set. The obtained results are shown favorable and encouraging for future work.

Keywords: Deep learning, tinyML, embedded systems, edge computing.

1. Introducción

En la actualidad, el crecimiento de la población junto con la migración de personas del campo hacia áreas urbanas ha provocado un aumento en la demanda de productos alimenticios. De acuerdo con el Instituto Nacional de Estadística y Geografía (INEGI),

Tabla 1. Estado del arte de sistemas TinyML aplicados a la agricultura.

Ref	Modelo	Dataset	Exactitud	Dispositivo de borde	Objetivo
[6]	CNN + LSTM	Plan Village	97.18 %	Servidor de borde	Predicción de enfermedades de frutas.
[21]	MLP	Propio	99 %	Raspberry Pi 3B	Inspección de la naraja.
[9]	CNN	Propio	97.39 %	ESP32-S3 (MCU)	Proceso post-cosecha de la fruta del olivo.
[13]	MobileNet	VegFru	82.72 %	Teléfono móvil	La clasificación de imágenes de frutas y verduras.
[7]	Inception	DiaMOS	99.73 %	ESP32-CAM	Identificación de enfermedades en la pera.

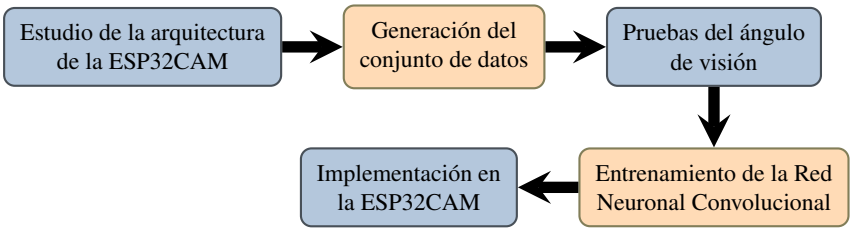


Fig. 1. Etapas de implementación de solución.

en México el porcentaje de población en áreas urbanas creció del 43 % en 1950 al 79 % en el 2020. Por otro lado, en 1950 la población rural constituia el 57 % y para el 2020 se ubicaba en 21 % [11]. Esta migración ha reducido el número de personas que trabajan en las zonas rurales, lo que ha provocado una escasez de mano de obra y una menor producción agrícola, lo que ha propiciado el uso de técnicas de cultivo sin suelo, cómo son la aereoponia y la hidropónia [3].

El uso de dichas técnicas ha facilitado el cultivo en regiones donde las prácticas agrícolas tradicionales son complicadas debido a condiciones climáticas o del suelo dando como resultado una mejora en la cantidad de productos hortícolas, lo que indica un futuro prometedor para la agricultura urbana [5]. Además, estas técnicas de cultivo sin suelo han sido reconocidas por su capacidad para mejorar la eficiencia en el uso de nutrientes y la calidad de los cultivos [24].

La agricultura sin suelo se encamina hacia una automatización integral, abarcando desde la germinación de semillas hasta la cosecha. Esta transformación se impulsa por la aplicación de inteligencia artificial, visión artificial, aprendizaje profundo y sistemas de monitoreo, optimizando cada fase del proceso productivo [22]. En este contexto, TinyML emerge como una tecnología clave para la agricultura de precisión porque ayuda al análisis e inferencias de datos localmente, reduciendo la latencia, mejorando la seguridad y la privacidad, y al ahorro del ancho de banda al evitar la necesidad de enviar datos a servidores centralizados para su procesamiento.

De esta forma, TinyML describe el uso de modelos de aprendizaje automático (ML, por sus siglas en inglés) para dispositivos periféricos o unidades de microcontrolador (MCU) con capacidades computacionales restringidas [17]. Con la ayuda de este enfoque los algoritmos de ML se pueden implementar y entrenar en dispositivos compactos de bajo consumo, lo que permite el procesamiento de datos en tiempo real y la toma de decisiones en el borde de la red. Por lo anterior, en este artículo se presenta un sistema de detección de pimientos morrón enfocado a la agricultura sin suelo, basado en dispositivos de borde y TinyML.

(a) Sub conjunto de pimientos normalizado. (b) Sub conjunto de no-pimiento normalizado.

Fig. 2. Conjunto de datos redimensionado.

(a) Imagen de 96×96 a 18 cm. (b) Imagen de 96×96 a 24 cm. (c) Imagen de 160×120 a 18 cm.

Fig. 3. Capturas realizadas sin flash de la ESP32CAM.

En contraste con los enfoques tradicionales donde se requieren servidores de gran capacidad computacional para ejecutar modelos de aprendizaje profundo complejos, nuestro sistema propone una solución de bajo costo y eficiente. La propuesta consiste en un dispositivo ESP32CAM que implementa una red neuronal MobileNet entrenada para detectar pimientos morrones en imágenes. El entrenamiento se realizó con un conjunto de datos de 1910 imágenes, principalmente obtenidas de internet. Los resultados obtenidos demuestran el potencial de TinyML para contribuir a la seguridad alimentaria, la eficiencia en el uso de recursos y la sostenibilidad de la agricultura en áreas urbanas.

2. Antecedentes

En la literatura existen diversos trabajos donde se hace uso de las redes neuronales y el aprendizaje profundo para el monitoreo del pimiento morron en diversas etapas. Ejemplos de ello son [15, 14] donde utilizan modelos de redes neuronales convolucionales para la identificación de enfermedades de pimiento morrón a través de la clasificación de sus hojas. Para esto, utilizaron una versión reducida el conjunto de datos Plant Village donde sólo se centran en las hojas de pimiento morrón.

(a) Imagen de 160×120 a 24 cm. (b) Imagen de 240×176 a 18 cm. (c) Imagen de 176×144 a 24 cm.

Fig. 4. Capturas realizadas utilizando ESP32CAM.

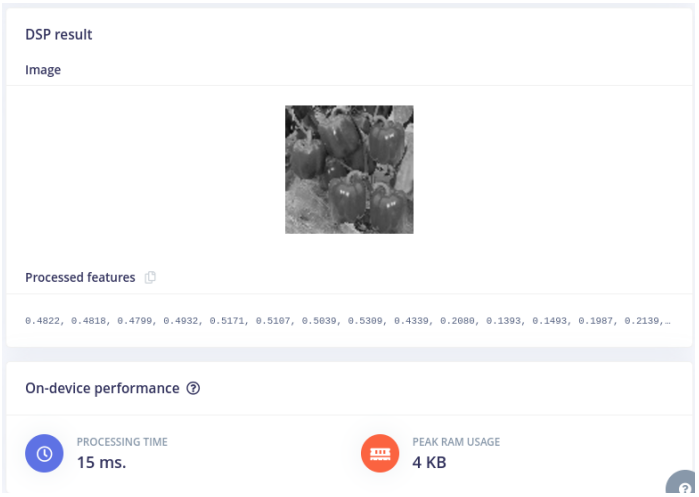
[15] utilizando una arquitectura arquitecturas VGG19 obtuvo un una exactitud del 94.68 %, mientras que [14] utilizando una red neuronal convolucional (ConvNet) de 3 capas convolucionales obtuvo un 96.88 %. Por otro lado, [2] utilizando un conjunto de datos de 2368 imágenes de pimiento morrón y mediante una red neuronal convolucional (ConvNet) clasifica en 3 tipos: pimiento verde, pimiento rojo y pimiento amarillo; logrando un 100 % de exactitud.

No obstante, los enfoques tradicionales emplean modelos neuronales computacionalmente demandantes no optimos para dispositivos de baja potencia. Por lo cual, TinyML utiliza el diseño de modelos de aprendizaje (profundos y no profundos) adecuados para dispositivos de limitado poder computacional. Investigaciones recientes se han centrado en la aplicación de la IA en dispositivos embebidos con el objetivo de implementar modelos de ML en dispositivos con recursos limitados con diversos objetivos [12].

Un ejemplo de ello es [16], donde destaca la importancia de incorporar IA en los dispositivos de IoT mediante la implementación de modelos de ML en hardware de bajo rendimiento. De la misma manera, [16] examina los modelos, estructuras y criterios para integrar el aprendizaje automático de vanguardia en dispositivos de IoT. De esta manera, la IA integrada en dispositivos con recursos limitados ha encontrado numerosas aplicaciones en diversos campos, incluidos la visión por computadora, la atención médica, la robótica, entre otras [1]. Asimismo, revisiones sistemáticas han sintetizado los avances existentes sobre este campo destacando el progreso, los desafíos y trazando una posible ruta futura [8, 19].

En general, la investigación en aplicaciones de IA en dispositivos con recursos limitados abarca varios dominios, desde sensores de IoT energéticamente eficientes hasta la evaluación comparativa de modelos de detección de objetos en dispositivos integrados, mostrando las diversas aplicaciones y avances en el campo. TinyML ha demostrado ser particularmente útil en el seguimiento de hortalizas, monitoreo de sistemas hidropónicos y en general en la agricultura de precisión [23].

Un ejemplo de ello es en [9] donde se presenta una implementación de una red neuronal convolucional, la cual contiene dos capas convolucionales, dos capas de normalización batch y una capa densa, dentro de un dispositivo ESP32-S3 para la inspección post-cosecha del fruto del olivo de la variedad Jordania. Dicha implementación logra una exactitud de 97.39 % utilizando imágenes de 50×50 pixeles.



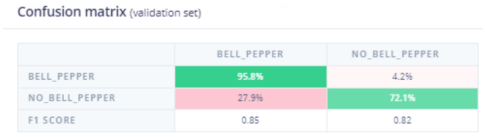
(a) Apartado para configurar los parámetros de las imágenes.

(b) Explorador de características y rendimiento.

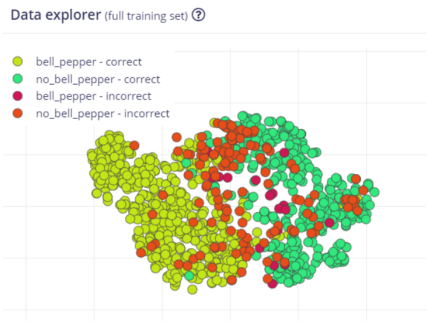
Fig. 5. Bloque de procesamiento de edge impulse.

Por otro lado, [7] donde presenta Lite-Agro, un sistema de identificación de enfermedades en la pera, a través del análisis de imágenes de hojas de pera utilizando TinyML. Para este trabajo se implementó una red neuronal profunda con arquitectura de tipo InceptionV3, inspirada en una arquitectura Xception, el cual logra una exactitud del 99.73 %. Dicho modelo fue entrenado durante 100 epochas utilizando el dataset DiaMOS, el cual contiene 3505 imágenes de hojas de pera divididas en cuatro clases. Finalmente el modelo se implementó en un dispositivo ESP32-CAM.

(a) Precision y valor de pérdida.



(b) Matriz de confusión.



(c) Gráfica de resultados al clasificar las imágenes de entrenamiento.

(d) Resultados de entrenamiento del modelo.

Fig. 6. Resultados del bloque de entranamiento de edge impulse.

En el mismo contexto de TinyML, existen otros trabajos que han empleado redes neuronales para el proceso de inspección y detección de enfermedades en plantas, en su gran mayoría han optado por dispositivos con baja potencia computacional diferentes a microcontroladores como teléfonos móviles y Sistemas en un chip (SoC, del inglés system on a chip) [6, 21, 13]. En la Tabla 1 se presenta un breve resumen del estado del arte de sistemas TinyML aplicadoa a la agricultura.

3. Metodología

El objetivo de este trabajo es detectar la existencia, o no existencia, de pimientos morrón en una imagen mediante una red neuronal convolucional (CNN). La metodología utilizada se divide en cinco etapas: (1) Estudio de la arquitectura de la ESP32CAM, (2) Generación del conjunto de datos, (3) Pruebas de ángulo de visión de la ESP32CAM, (4) Entrenamiento del modelo de Red Neuronal Convolucional (5) Implementación del modelo de aprendizaje en la ESP32CAM. Las etapas realizadas en este trabajo se ilustran en la Figura 1. Para la realización de este trabajo se utilizaron las siguientes herramientas y recursos: el IDE de Arduino, TensorFlow Lite Micro, el sistema Edge Impulse [20] para el diseño y entrenamiento de la red neuronal. Asimismo, la evaluación del sistema se realizó mediante pruebas de rendimiento para determinar la precisión, la sensibilidad y la velocidad de la red neuronal entrenada.

3.1. Arquitectura de la ESP32CAM

La ESP32CAM es un módulo de bajo costo y bajo consumo de energía el cual integra un microcontrolador ESP32 con un sensor de cámara OV2640.

Tabla 2. Hiperparámetros para el entrenamiento del modelo.

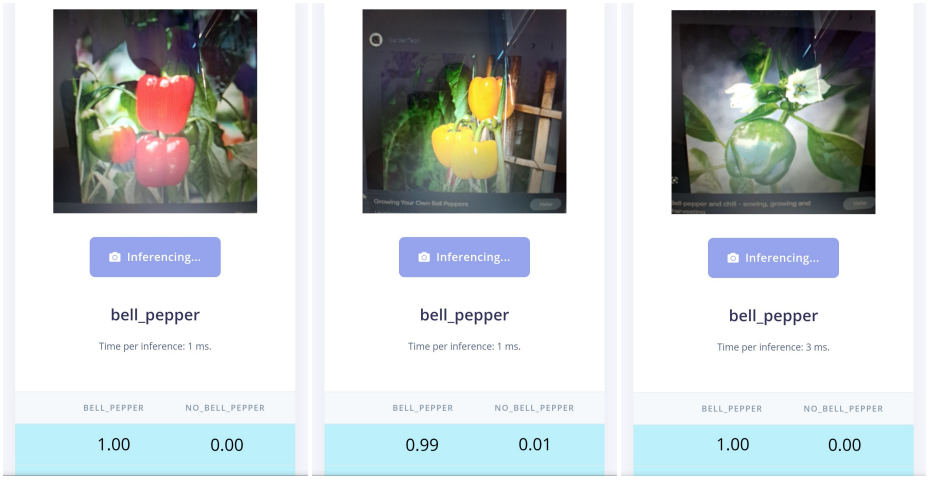
Modelo	lr	MN alpha	Densa	Dropout	exac.	f1	prec.	sens.
MovileNetV2	0.01	0.05	32	0.3	77.55	0.8	0.84	0.77
MovileNetV2	0.001	0.05	32	0.3	81.46	0.84	0.88	0.8
MovileNetV1	0.001	0.25	32	0.1	90.86	0.92	0.95	0.9
MovileNetV1	**0.01**	**0.25**	**16**	**0.1**	**91.12**	**0.93**	**0.95**	**0.91**
MovileNetV1	0.01	0.2	32	0.1	90.08	0.91	0.94	0.89
MovileNetV1	0.001	0.1	32	0.1	83.29	0.87	0.89	0.85
MovileNetV2	0.01	0.35	32	0.1	87.73	0.86	0.93	0.8
MovileNetV2	0.001	0.35	32	0.1	89.56	0.89	0.94	0.84
MovileNetV2	0.001	0.35	32	0.3	87.73	0.89	0.94	0.84

Tiene un microcontrolador de dos núcleos LX6 de 32 bits que operan a una frecuencia de hasta 240 MHz. Tiene una capacidad de 4MB de memoria flash y 520 KB de SRAM. La ESP32CAM cuenta con WiFi, Bluetooth y BLE integrados que permiten la comunicación inalámbrica con otros dispositivos, soporta cámaras OV2640 y OV7670 que puede capturar imágenes de hasta 2 megapixeles [4]. Esta arquitectura la convierte en una plataforma conveniente y de muy bajo costo para la implementación de aplicaciones de visión artificial en el contexto de TinyML.

3.2. Conjunto de datos

Para este trabajo se utilizó un conjunto de datos con las clases "pimiento" y "no-pimiento", dicho conjunto de datos se obtuvo mediante la unión de dos conjuntos de datos públicos de pimiento morrones: el primero se obtuvo de [18], el cual es un conjunto de datos con 3825 imágenes para el reconocimiento de diversas frutas y verduras; el segundo se obtuvo de [10], el cual contiene 1300 imágenes etiquetadas de pimientos morrón. Adicionalmente al conjunto de datos se agregaron imágenes de diversos tipos de árboles, plantas cultivadas en macetas, etc, las cuales se descargaron manualmente de Google. Algunas de las imágenes en el conjunto de datos se muestra en la figura 2. Debido a que las imágenes obtenidas de diversas fuentes tienen distintas dimensiones, éstas se redimensionaron a 96×96 mediante un Script de python.

Posteriormente para el caso de la clase "pimiento", se eliminaron de manera manual imágenes que no correspondiesen íntegramente a pimientos. El conjunto de datos final contiene 1910 imágenes, donde 876 pertencen a la clase "pimiento" y 1034 pertencen a la clase "no-pimiento". Para la etapa de entrenamiento, ver sección 4, el conjunto de datos se dividió en una proporción del 80 % para entrenamiento (1527 imágenes) y el 20 % restante (383 imágenes) para prueba. Sin embargo, y debido a las limitaciones que impone Edge Impulse, no se utilizó una validación cruzada para el caso particular de este trabajo.



(a) Primera prueba. (b) Segunda prueba. (c) Tercera prueba.

Fig. 7. Pruebas en Edge Impulse de la clase “pimiento”.

3.3. Pruebas de ángulo de visión de ESP32CAM

Durante las pruebas iniciales de la ESP32CAM se observó que el ángulo de visión cambia en función de la resolución de ésta, además no cuenta con un sistema de auto-foco por lo que se deberá tener contemplado la resolución de las imágenes a obtener para posicionar el dispositivo a una distancia tal que se logre capturar el fruto en su totalidad y lo mas claro posible. Se realizaron capturas con resoluciones de 96×96, 160×120, 176×144 y 240×176, debido a que el dispositivo ESP32CAM no posee recursos computacionales necesarios para almacenar una red neuronal que clasifique imágenes con mayor resolución. Además, se capturaron a una distancia de 18 y 24 centímetros del objetivo; ésto para realizar una comparación entre las imágenes tomadas a estas dos distancias y poder optar por la más adecuada para una implementación en un sistema real a futuro.

En la figura 3 y en la figura 4 se pueden apreciar las diferencias entre las imágenes capturadas a diferente distancia, también se logra notar cómo el ángulo de visión de las imágenes aumenta al cambiar la resolución de 96×96 a 160×120 píxeles. Sin embargo, al aumentar la resolución a 176×144 aún se nota un ligero aumento en el ángulo de visión pero éste no es tan considerable como el anterior. Con respecto al flash en este caso no se observó ningúna variación debido a las las capturas se realizaron durante el día, no obstante, éste es un aspecto a tener en cuenta para una implementación real a futuro.

4. Entrenamiento de modelo en Edge impulse

En este proyecto se optó por el uso de la plataforma Edge Impulse [20], la cual es una plataforma que facilita el desarrollo e implementación de modelos de aprendizaje automático en dispositivos de borde, como microcontroladores, sensores y FPGAs.

(a) Primera prueba. (b) Segunda prueba. (c) Tercera prueba.

Fig. 8. Pruebas en la ESP32CAM de la clase “pimiento”.

Entre las características que ofrece se encuentran la captura, etiquetado y preparación de datos, selección, entrenamiento y optimización de modelos de aprendizaje. Una vez creado el proyecto en Edge Impulse, y subir el conjunto de imágenes, se procedió a generar un Impuso (Impulse) el cual toma datos crudos, extrae las características y luego usa un bloque de aprendizaje para clasificar nuevos datos. El impulso esta compuesto por 3 secciones principales las cuales se listan a continuación.

– Bloque de entrada (Image data) determina el formato de los datos con los que se está entrenando el modelo.

– Bloque de procesamiento (Processing block) es el bloque donde se realizan la extracción de características y el preprocesamiento de datos.

– Bloque de aprendizaje (Learning block) realiza el proceso de entrenamiento del modelo que se seleccionó, existiendo diversos algoritmos de aprendizaje que incluyen tareas para clasificación, regresión, detección de anomalías, transferencia de imágenes, detección de palabras clave o detección de objetos.

El bloque de procesamiento se configura la transformación de las imágenes en escala de grises, ya que ésto hace que la red neuronal utilice menos recursos a la hora de ser implementada en la ESP32CAM, tal como se muestra en la Figura 5a. Una vez ejecutado el bloque de procesamiento, se muestra el tiempo de ejecución y la cantidad máxima de memoria RAM utilizada además de una comparación del espacio de clasificación de las clases, tal como se muestra en la figura 5b.

En el bloque de aprendizaje se configura la arquitectura de la red neuronal, además de agregar los valores de diversos hiperparámetros para el aprendizaje tales como el número de épocas, la tasa de aprendizaje, el porcentaje del subconjunto de validación, entre otros. Al finalizar el entrenamiento del modelo, Edge Impulse muestra los resultados de precisión, pérdida, matriz de confusión y una gráfica que representa los resultados al clasificar las imágenes del dataset; además presenta el rendimiento del modelo (Figura 6).

(a) Primera prueba. (b) Segunda prueba. (c) Tercera prueba.

Fig. 9. Pruebas en la ESP32CAM de la clase "no-pimiento".

4.1. Implementación de modelo en ESP32CAM

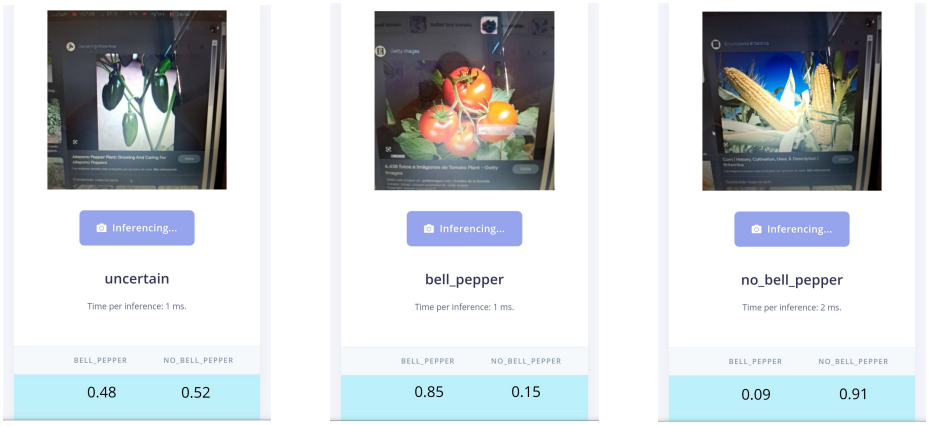
El modelo entrenado se implementó en la ESP32CAM utilizando la plataforma Edge Impulse. En primer lugar, el modelo se exportó como una librería de Arduino, la cual contiene el modelo de la CNN en formato hexadecimal, permitiendo su ejecución en el dispositivo mediante TensorFlow Lite Micro. En segundo lugar se modificó la librería para que la ESP32CAM funcione como un servidor web permitiendo recibir solicitudes HTTP desde un navegador web externo, y poder enviar la imagen capturada de regreso. En resumen, la implementación del modelo en la ESP32CAM se realizó exitosamente, permitiendo la interacción con la cámara y el modelo de clasificación a través de un servidor web.

5. Resultados experimentales

Para este trabajo se realizaron diversos experimentos donde se utilizó la arquitectura de red neuronal convolucional de MobileNet. Se optó por el uso de la arquitectura MobileNet ya que esta diseñada específicamente para ejecutarse en dispositivos con recursos limitados, además de que existen modelos preentrenados con conjuntos de datos como COCO2017, lo que permite disminuir el tiempo de entrenamiento mediante el proceso de entrenamiento de la red. De este modo, cada modelo fue entrenado durante 100 épocas, utilizando un batch size de 32 elementos.

Por cada experimento se varió de manera manual los hiperparámetros de tasa de aprendizaje, versión de la arquitectura MobileNet, el valor del peso alpha de MobileNet, cantidad de neuronas en la capa densa y el porcentaje de dropout. Se estableció una relación $80/20$ para los conjuntos de entrenamiento y prueba. Adicionalmente, se fijó un $20\,\%$ para validación durante el entrenamiento.

Las dos versiones de MobileNet (versión 1, y Versión 2) utilizadas por Edge Impulse están preentrenadas con el conjunto de datos de COCO2017, de esta manera se utiliza la técnica de transferencia de conocimiento (Transfer learning) para fijar los valores de los pesos de las capas convolucionales y sólo optimizar los valores de los pesos en la capa densa.



(a) Primera prueba. (b) Segunda prueba. (c) Tercera prueba.

Fig. 10. Pruebas en Edge Impulse de la clase “no-pimiento”.

Ésto da como consecuencia una disminución en las épocas necesarias y el tiempo de entrenamiento. Finalmente el modelo es quantizado a Int8 para disminuir su tamaño y pueda ser implementado en dispositivos de borde. En la tabla 2 se proporcionan la configuración para el entrenamiento de cada modelo y los resultados obtenidos con el subconjunto de prueba.

5.1. Pruebas mediante Edge impulse

Edge Impulse ofrece una manera sencilla de poner a prueba el modelo a través del navegador web. Inicialmente se probó mediante imágenes de varios objetos que se podrían encontrar en un huerto como herramientas de jardinería, plantas sin ningún fruto, insectos, etc, así como imágenes de pimientos morrones, que son los que se busca identificar. En la figura 7 se muestran algunas pruebas del modelo mediante la página de Edge Impulse para la clase “pimiento”.

Como se puede observar se logra identificar imágenes donde existe presencia de pimientos morrón. En general el modelo aún clasifica de manera incorrecta imágenes de frutos similares a pimientos, hojas de arboles e insectos parados en hojas. En la figura **??**, se puede observar que aúnque el modelo clasifica correctamente muchas imágenes, existen falsos positivos. En la tabla 2 se proporcionan los valores de los hiperparámetros y los resultados obtenidos; se oberva una exactitud de $91{,}12\,\%$ del mejor modelo (ya quantizado a int8) hasta el momento.

5.2. Pruebas en la ESP32CAM

Luego de probar el modelo de clasificación dentro de la plataforma Edge Impulse, se implementó la red neuronal en la ESP32CAM siguiendo el procedimiento descrito en la sección 4.1. Posteriormente, se capturaron múltiples fotografías en el contexto del huerto de pimientos morrón. Algunos de los resultados de las clasificaciones realizadas por el modelo se presentan a continuación.

Estos resultados demuestran la capacidad del sistema para identificar y clasificar objetos en un entorno real, con un alto grado de precisión. En las figuras 8 y 9 se muestran algunos de los resultados de la detección de imágenes capturadas y clasificadas mediante la red neuronal implementada en la ESP32CAM. A pesar de que el sistema detecta la gran mayoría de pimientos morrones, aún se obtienen resultados falsos verdaderos como el que se observa en la figura 9c.

5.3. Discusión de resultados

Al haber realizado las pruebas del modelo tanto en la plataforma de Edge Impulse como con la ESP32CAM, se observó que el modelo entrenado es impreciso, especialmente a la hora de mostrarle otros frutos. No obstante, el sistema logra clasificar imágenes de pimientos morrones de forma correcta. Se sugiere que estos resultados son causa de una baja calidad del conjunto de datos, así como una carente optimización de los hiperparámetros en el entrenamiento del modelo.

De acuerdo a los resultados se pueden observar diversas ventajas: El modelo neuronal se implementó de forma exitosa en el dispositovo ESP32CAM, lo que demuestra la factivilidad del modelo para dispositivos de baja potencia, además dicho modelo muestra una alta exactitud en las muestras de pimientos morrones. Por parte de la ESP32CAM es un dispositivo de bajo costo y de fácil acceso, que conjunto con la plataforma Edge Impulse facilita la implementación de modelos para aplicaciones de TinyML. Por otro lado, se observaron diversas desventajas: En primer lugar, se debe seleccionar y/o implementar un modelo neuronal adecuado para cada dispositivo, ya que en ocasiones el modelo superaba las capacidades de memoría y almacenamiento de la ESP32CAM y no era posible la implementacióno bien, en ocasiones la ESP32CAM se congelaba. De igual manera, en ocasiones el tiempo de respuesta superaba los 5 segundos, que para algunas aplicaciones puede ser un tiempo importante.

Como trabajos futuros se enfocará en mejorar la precisión del modelo, ampliar su funcionalidad y explorar diferentes plataformas de hardware, debido a que la ESP32CAM impone importantes limitaciones como son: a) la admición de redes neuronales convolucionales ligeras como la MobileNet v2 con un máximo de 32 neuronas, b) sólo procesa imágenes en escala de grises con una resolución limitada de 96×96 píxeles. Estas limitaciones hacen todo un reto para la implementación de proyectos en ambientes reales. Tomando en cuenta dichas limitaciones y los resultados obtenidos en este trabajo, se proponen las siguientes líneas de investigación a futuro: I) Expandir la capacidad del modelo para clasificar el nivel de maduración de los pimientos. II) Realizar una comparación del rendimiento de diferentes dispositivos de borde de bajo costo como ESP-Eye, Raspberry Pi Pico y Arduino 33 BLE Sense. III) Evaluar el rendimiento y la precisión de diferentes arquitecturas de redes neuronales "ligeras" como SqueezeNet, MobileNet y ShuffleNet.

6. Conclusiones

En el presente trabajo se realizó un estudio de la arquitectura y el funcionamiento de la ESP32CAM para la detección de pimientos en imágenes mediante una red neuronal MobileNet.

El modelo obtuvo una exactitud de $91{,}12\,\%$ logrando identificar en su mayoría imágenes con pimientos estando presentes, sin embargo, se observan dificultades en la clasificación de objetos como herramientas de jardinería y otros frutos distintos a pimientos morrones. Ésto se le puede atribuir a las limitaciones de la ESP32CAM, como su capacidad limitada para almacenar redes neuronales y procesar imágenes, pero principalmente por la calidad del dataset. Sin embargo, es importante destacar que el modelo no presentó falsos negativos al identificar pimientos morrones, lo que constituye un logro significativo para el desarrollo de un sistema de clasificación de pimientos morrones basado en dispositivos de borde de bajo costo. Los resultados mostrados en este trabajo alentadores para la implementación de un sistema de monitoreo autónomo y de bajo costo para sistemas agricolas urbanos y/o sin suelo.

Referencias

1. Ajani, T. S., Imoize, A. L., Atayero, A. A.: An overview of machine learning within embedded and mobile devices–optimizations and applications. Sensors, vol. 21, no. 13, pp. 4412 (2021) doi: 10.3390/s21134412
2. Almadhoun, H. R.: Bell pepper classification using deep learning. International Journal of Academic Engineering Research, vol. 5, no. 1, pp. 75–79 (2021)
3. Bassie, H., Sirany, T., Alemu, B.: Rural-urban labor migration, remittances, and its effect on migrant-sending farm households: Northwest Ethiopia. Advances in Agriculture, vol. 2022, pp. 1–8 (2022) doi: 10.1155/2022/4035981
4. DFROBOT: Esp32-cam development board (2020) www.dfrobot.com/product-1879.html
5. Dhawi, F.: The role of plant growth-promoting microorganisms (PGPMs) and their feasibility in hydroponics and vertical farming. Metabolites, vol. 13, no. 2, pp. 247 (2023) doi: 10.3390/metabo13020247
6. Dhiman, P., Kaur, A., Hamid, Y., Alabdulkreem, E., Elmannai, H., Ababneh, N.: Smart disease detection system for citrus fruits using deep learning with edge computing. Sustainability, vol. 15, no. 5, pp. 4576 (2023) doi: 10.3390/su15054576
7. Dockendorf, C., Mitra, A., Mohanty, S. P., Kougianos, E.: Lite-agro: Exploring light-duty computing platforms for IoAT-edge AI in plant disease identification. Internet of Things Advances in Information and Communication Technology, pp. 371–380 (2023) doi: 10.1007/978-3-031-45882-8_25
8. Han, H., Siebert, J.: TinyML: A systematic review and synthesis of existing research. In: International Conference on Artificial Intelligence in Information and Communication, pp. 269–274 (2022) doi: 10.1109/ICAIIC54071.2022.9722636
9. Hayajneh, A. M., Batayneh, S., Alzoubi, E., Alwedyan, M.: TinyML olive fruit variety classification by means of convolutional neural networks on IoT edge devices. AgriEngineering, vol. 5, no. 4, pp. 2266–2283 (2023) doi: 10.3390/agriengineering5040139
10. images.cv: 1.3k bell pepper labeled image dataset (2020) images.cv
11. Instituto Nacional de Estadística y Geografía: Población rural y urbana (2020) cuentame.inegi.org.mx/poblacion/rur_urb.aspx?tema=P
12. Kwon, J., Park, D.: Hardware/software co-design for TinyML voice-recognition application on resource frugal edge devices. Applied Sciences, vol. 11, no. 22, pp. 11073 (2021) doi: 10.3390/app112211073
13. Liu, C., Wang, X., Ni, J., Cao, Y., Liu, B.: An edge computing visual system for vegetable categorization. In: Proceedings of the 18th IEEE International Conference On Machine Learning And Applications, pp. 625–632 (2019) doi: 10.1109/ICMLA.2019.00115

14. Mahamud, F., Neloy, M. A. I., Barua, P., Das, M., Nahar, N., Hossain, M. S., Andersson, K., Hoassain, M. S.: Bell pepper leaf disease classification using convolutional neural network. In: International Conference on Intelligent Computing and Optimization, vol. 569, pp. 75–86 (2022) doi: 10.1007/978-3-031-19958-5_8
15. Mathew, M. P., Elayidom, S., Jagathyraj, V.: Disease classification in bell pepper plants based on deep learning network architecture. In: Proceedings of the 2nd International Conference for Innovation in Technology, pp. 1–6 (2023) doi: 10.1109/INOCON57975.2023.10101269
16. Merenda, M., Porcaro, C., Iero, D.: Edge machine learning for AI-enabled IoT devices: A review. Sensors, vol. 20, no. 9, pp. 2533 (2020) doi: 10.3390/s20092533
17. Raza, W., Osman, A., Ferrini, F., Natale, F. D.: Energy-efficient inference on the edge exploiting TinyML capabilities for UAVs. Drones, vol. 5, no. 4, pp. 127 (2021) doi: 10.3390/drones5040127
18. Seth, K.: Fruits and vegetables image recognition dataset (2020) www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition
19. Shafique, M., Theocharides, T., Reddy, V. J., Murmann, B.: TinyML: Current progress, research challenges, and future roadmap. In: Proceedings of the 58th ACM/IEEE Design Automation Conference, pp. 1303–1306 (2021) doi: 10.1109/DAC18074.2021.9586232
20. Shelby, Z., Jongboom, J., Huang, S., Watkins, L., DeLey, W.: Edge impulse (2024) www.edgeimpulse.com
21. Silva, M., Ferreira-da-Silva, J., Oliveira, R.: IDiSSC: Edge-computing-based intelligent diagnosis support system for citrus inspection. In: Proceedings of the 23rd International Conference on Enterprise Information Systems, pp. 685–692 (2021) doi: 10.5220/0010444106850692
22. Susanto, F., Suryani, N. K., Darmawan, P., Prasiani, K., Ramayu, I. M. S.: Comprehensive review on automation in hydroponic agriculture using machine learning and IoT. RSF Conference Series: Engineering and Technology, vol. 1, no. 2, pp. 86–95 (2021) doi: 10.31098/cset.v1i2.479
23. Tang, Y., Chen, M., Wang, C., Luo, L., Li, J., Lian, G., Zou, X.: Recognition and localization methods for vision-based fruit picking robots: A review. Frontiers in Plant Science, vol. 11 (2020) doi: 10.3389/fpls.2020.00510
24. Tunio, M. H., Gao, J., Mohamed-Tarek, M. K., Ahmad, F., Abbas, I., Ali-Shaikh, S.: Comparison of nutrient use efficiency, antioxidant assay, and nutritional quality of butter-head lettuce (Lactuca sativa L.) in five cultivation systems. International Journal of Agricultural and Biological Engineering, vol. 16, no. 1, pp. 95–103 (2023) doi: 10.25165/j.ijabe.20231601.6794