UNIVERSIDAD NACIONAL TECNOLÓGICA DE LIMA SUR

FACULTAD DE INGENIERÍA Y GESTIÓN

ESCUELA PROFESIONAL DE INGENIERÍA SISTEMAS

Mejora de calidad en la selección de plátanos de seda utilizando la metodología Six Sigma, OpenCV y despliegue en Hugging Face Spaces con Gradio

Proyecto presentado para la asignatura de

Algoritmos de computación gráfica, dirigida por

Mg. Juan Carlos Reátegui Morales

Estudiantes

Andres Alfredo Rejas Ramirez

20b3010146

Edson Emanuel Alvarado Prieto

20b3110254

Jaime Giancarlo Matheus Guerra

20b3110064

Jairo Daniel Mendoza Torres

20b3010069

Juan Pablo David Rosado Valerio

20b3010135

Villa El Salvador, 2023

Índice

1. Introducción

1.1. Contexto y justificación

Este estudio se justifica por la necesidad de abordar las limitaciones existentes en los métodos tradicionales de selección de plátanos de seda y promover mejoras significativas en la calidad y eficiencia del proceso (Vilaboa, 1999; García et al., 2017). Este método consiste en utilizar un algoritmo de procesamiento de imágenes que extrae características visuales relevantes de los plátanos de seda, tales como el color, la forma, el tamaño y la presencia de defectos, se espera reducir los desperdicios, optimizar el proceso de selección y aumentar la satisfacción del cliente (Pereira et al., 2016). Con esta investigación, se pretende aportar al avance científico y tecnológico en el campo de la selección de alimentos, específicamente en el sector de plátanos de seda, ofreciendo a la industria alimentaria una solución práctica y aplicable para mejorar la calidad de los productos entregados, fortaleciendo su posición competitiva en el mercado global.

1.2. Objetivos de la investigación

Objetivo general:

- Mejorar la calidad en la selección de plátanos de seda utilizando la metodología Six Sigma y OpenCV para la obtención de datos

Objetivos específicos

- Desarrollar un proyecto en lenguaje Python para la identificación de madurez de plátanos de seda..
- Identificar defectos en el proceso y proponer mejoras para la calidad.

1.3. Antecedentes

Montalvo et al. (2017) implementaron la metodología Six Sigma en la producción de alimentos en una empresa productora de huevo. El desarrollo de esta metodología comienza con la definición del problema, en el cual se establece desde dónde se tiene que iniciar para brindar una solución. Seguido de ello, en la etapa de medición se trabajó con una muestra de los alimentos, identificando la variabilidad existente en los cambios de estos. Posteriormente, durante el análisis, se determinó que el problema era la falta de planeación en el área de producción. Esto dio paso a la mejora, en la cual se determinaron las propuestas para mejorar la producción de alimentos y los días de cambios, teniendo en cuenta la edad de cada

parvada. Finalmente, en la etapa de control se estableció procurar mantener las mejoras aplicadas, con el fin de evitar futuros errores.

Tello & Aguirre (2019) proponen la implementación de una estrategia de negocios basada en Six Sigma, con el fin de mejorar la calidad de los productos. En su artículo, mencionan las ventajas de dicha implementación, siendo estas el retorno financiero rentable y la reducción de costos, mediante la mejora de calidad de un producto seleccionado. Esta mejora consiste en la reducción de errores o defectos y de tiempo de producción o ciclo. Asimismo, se menciona que, como requisito para su implementación en una empresa, esta debe contar con conocimientos de métodos estadísticos y velar por el bienestar de los empleados y recursos humanos materiales. Cabe mencionar que, en su artículo, se afirma que la metodología Six Sigma puede resultar muy beneficiosa en el contexto de las PYMES, debido a la mayor competitividad entre estas, es decir, se mejoraría la capacidad de reacción ante cambios constantes presentes en su entorno, manteniendo o incluso mejorando la calidad exigida por el consumidor.

De acuerdo con los autores Piedad, Larada, et al. (2018), el empleo de la imagen hiperespectral en el ámbito del control de calidad y seguridad alimentaria es una técnica altamente relevante que puede proveer valiosa información para el presente proyecto. En efecto, los autores explican que la imagen hiperespectral consiste en una técnica de análisis de imágenes que permite identificar características específicas de los alimentos, tales como su composición química, textura y color. Además, el estudio en cuestión describe cómo esta técnica ha sido empleada para detectar contaminantes en los alimentos, como bacterias y metales pesados, así como para evaluar la calidad de estos, incluyendo su frescura y madurez (Piedad, Larada, et al., 2018).

Cachique Maldonado, F. H. (2022) en su tesis “Propuesta de Implementación de la metodología Six-Sigma, para mejorar el control de calidad, en el proceso de recubrimiento en el área de pintura, en la empresa Fabertek SAC” impedimenta la metodología Seis Sigma. En la fase definir, se busca mejorar los controles de calidad por lo que se establece los alcances y limitaciones que deben tener cada miembro del equipo. En la fase medir, se establece la muestra a utilizar para determinar donde se encuentra la mayor ocurrencia de defectos en el proceso de recubrimiento en el área de pintura. En la fase analizar, se establece cuáles son las causas que originan los elementos defectuosos. En la fase mejorar, se detalla el mejor proceso de recubrimiento en el área de pintura con la finalidad de disminuir los elementos

defectuosos, llegando a tener una calidad del producto final. Por último, en la fase controlar, se evalúa la aplicación de la mejora en el proceso de recubrimiento, comparando los datos anteriores y los actuales.

Luna Amador, J. A. (2021), hizo un estudio acerca de la implementación de la metodología Sigma Seis en el proceso de manejo de los equipajes de una aerolínea estadounidense que opera en la ciudad de Cartagena - Colombia. Para el desarrollo de la investigación se basó en descripción en detalle del proceso, midiendo el desempeño de los subprocesos que consideraron como críticos, para después analizarlos con AMEF y concluir estableciendo mejoras y mecanismos de prevención de fallas, logrando obtener como resultados que el proceso actual es bueno, cuando solo se trata de un vuelo, es por ello que debido a que la meta de su investigación es tener "cero errores" siempre, es necesario buscar mecanismos de prevención con la finalidad de minimizar al máximo los riesgos de fallas, para de esta manera mejorar los indicadores de calidad del proceso.

Jiménez Velasco et. al (2018) en su tesina "Aplicación de la metodología Seis Sigma para mejorar el proceso de ventas en la empresa TIPITOP S.A.P.I. DE C.V." se implementó las cinco etapas de Seis Sigma: definir, medir, analizar, mejorar y controlar para realizar mejoras en los procesos de la empresa TIPITOP. Los procesos no estaban estandarizados y las ventas se realizaban sin ningún protocolo vendedor-comprador -de forma empírica-. Además de otros defectos como un personal no capacitado, pocos apartados electríonicos y la baja Gama de Vehículos. En ese sentido, se utilizó la metodología Seis Sigma para redefinir el objetivo de ventas mensual, utilizar un indicador para identificar si se ha alcanzado la meta, agregar una aplicación de ventas para complementar la web -así como la competencia-, establecer mayores sucursales para tener más presencia en el mercado y controlar cada una de las etapas dentro de los procesos optimizados.

Dominguez Ramirez, W. (2019), en su tesis "Aplicación de la metodología Six Sigma para mejorar el proceso de picking de una empresa de alimentos Callao-2019" implementa las cinco etapas de la metodología Seis Sigma, en la etapa de definición se encontró un defecto en el proceso de picking; la etapa de medir se realizó una medición sobre el índice de rechazos de mercadería por parte de los clientes; la etapa de analizar se realizó una medición de la frecuencia con que se cometen las incidencias para entender el intervalo de tiempo en el cúal se cometen errores; la etapa de mejorar se identificó el área donde ocurren mayores incidencias y se atendió el problema; y finalmente en la etapa de controlar se hizo

seguimiento sobre la importancia del cumplimiento de los controles. De este modo, según los gráficos estadísticos obtenidos de la empresa se concluyó que la metodología Six sigma mejora el proceso de picking.

Escobar & Roa (2016) desarrollaron una herramienta computacional que utilizaba los algoritmos de la librería OpenCV para identificar el estado de madurez de las granadillas. La técnica utilizada fue Clustering K-medias, la cual permitió la validación de las imágenes de granadillas de distinto color. Dicha herramienta resultó tener un 93% de porcentaje de acierto, evidenciado por las comparaciones realizadas por un técnico experto. Los problemas que surgieron durante el proceso principalmente se debieron a la iluminación, por lo que propusieron trabajar con parámetros de iluminación controlados para capturar las imágenes.

2. FUNDAMENTOS TEÓRICOS

2.1. Metodología Six Sigma

La metodología Seis Sigma es una estrategia de negocios de mejora continua, en el cual toma como enfoque la satisfacción del cliente, al mejorar el desempeño y la entrega de producto, en un manejo de metodologías y diseños, que facilita eliminar los cambios en los procesos y disminuye las deficiencias utilizando herramientas estadísticas. (Rubio, R. 2016)

El sigma es un símbolo en la estadística y representa a la desviación estándar, el cual describe la variación que hay en un grupo de datos

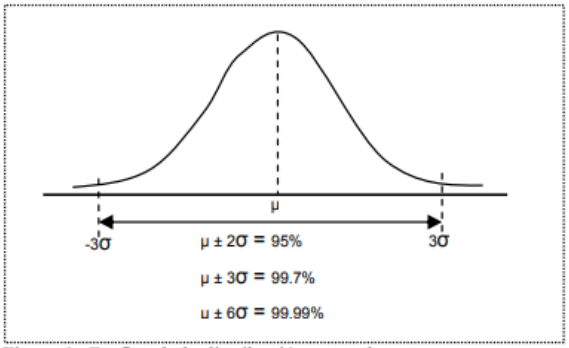


Figura 1. Grafica de la distribución normal

2.2. Etapas de la metodología de Six Sigma

2.2.1. Etapas 1: Definir

Es la fase genérica, en esta etapa se va conociendo los procesos, actividades, las personas que trabajan, para encontrar el problema o la oportunidad de mejora, se establecen los objetivos, que buscan satisfacer los requisitos del cliente en todo su proceso, una herramienta el el benchmarking, que compara los procesos con los de las empresas líderes en el mercado. (Rodrigo Oltra y Gisbert Soler, V.; 2016).

2.2.2. Etapas 2: Medir

Según Rodrigo Oltra y Gisbert Soler, V. (2016). Con los problemas identificados en los procesos internos que afectan el producto final o los servicios, en la etapa anterior, se busca utilizar herramientas para la recolección y análisis de datos. A la hora de medir, se utilizan estudios de capacidad de proceso y correlación entre defectos y confiabilidad. Además, otras herramientas que se emplean son el diagrama de flujos de proceso, el histograma y los diagramas de tendencias.

2.2.3. Etapas 3: Analizar

El propósito es comprender las causas de los defectos mediante técnicas como tormentas de ideas y herramientas estadísticas, que identifican variables clave y examinan los resultados óptimos para estandarizar los procedimientos. Durante la fase de análisis, se utilizan técnicas RCA, como el Diagrama de Pareto para identificar las principales causas de los problemas y los Diagramas de Causa-Efecto para llegar a las causas raíces. Además, se emplean Diagramas de Dispersión para relacionar dos variables y hacer estimaciones de manera rápida. (Rodrigo Oltra y Gisbert Soler, 2016)

2.2.4. Etapas 4: Mejorar

La fase de mejora tiene como objetivo identificar las variables que se pueden mejorar para cuantificar su efecto sobre las características más críticas de la calidad, y en base a su relevancia, mejorar el proceso para cumplir con los márgenes aceptables. Durante esta fase se desarrollan todas las estrategias de mejora, definiendo los factores que se controlarán para medir el efecto sobre las características críticas y planificando la mejor forma de llevar a cabo la mejora, buscando el desempeño óptimo del proceso. (Rodrigo Oltra y Gisbert Soler, 2016)

2.2.5. Etapas 5: Controlar

Durante esta última etapa se busca cumplir que las modificaciones en las variables estén dentro de los límites de variación permitidos, para lo cual emplean técnicas como el Control Estadístico de Procesos y gráficas de control, Además, se utiliza la técnica lean Poka Yoke, entre otras, para garantizar que no se produzcan errores. De esta forma, se establece un proceso de mejora continua. (Rodrigo Oltra y Gisbert Soler, 2016)

2.3. OpenCV: Visión por computadora y procesamiento de imágenes

2.3.1. Conceptos básicos de OpenCV

La página oficial de OpenCV destaca que es una biblioteca de código abierto que ofrece una gran variedad de algoritmos de visión artificial. La biblioteca se organiza en módulos, lo que significa que el paquete contiene varias bibliotecas dinámicas o estáticas. Los módulos disponibles son: Funciones básicas (core), Procesamiento de imágenes (imgproc), Análisis de video (video), Calibración de cámara y reconstrucción 3D (calib3d), Extracción de características 2D (features2d), Detección de objetos (objdetect), Interfaz gráfica de usuario (highgui) y Entrada/salida de video (videoio). Además, hay otros módulos

auxiliares, como FLANN y Google test wrappers, enlaces de Python y otros. (Doxygen, 2022)

2.3.2. Procesamiento de imágenes con OpenCV

De acuerdo con Bytepeaker (2021), la biblioteca OpenCV permite llevar a cabo operaciones básicas en imágenes, tales como la lectura y visualización de imágenes, la modificación del tamaño y rotación de imágenes, la conversión a escalas de grises o binarias, la aplicación de filtros y máscaras, la detección de bordes y contornos, el reconocimiento de formas geométricas y colores, y el uso de la transformada de Hough y el modelo Pinhole. Asimismo, se puede reducir o aumentar el tamaño de una imagen con la ayuda de la biblioteca OpenCV. Por otra parte, William (2017) indica que el procesamiento de imágenes puede mejorar su precisión mientras se aumente el número de imágenes utilizadas en el entrenamiento del modelo de inteligencia artificial para el reconocimiento de imágenes.

2.3.3. Aplicaciones en la clasificación de plátanos de seda

OpenCV, en conjunto con técnicas de visión por computadora, ha demostrado ser una herramienta eficaz en la clasificación de plátanos de seda. La aplicación de OpenCV en este campo ha permitido el desarrollo de algoritmos y sistemas capaces de identificar características de madurez, calidad y defectos en los plátanos de seda. Diversos estudios han explorado la implementación de OpenCV en la clasificación de plátanos de seda. Por ejemplo, García et al. (2017) utilizaron técnicas de procesamiento de imágenes y análisis de características visuales para detectar agentes patógenos en el cultivo de plátano. Sus resultados demostraron la viabilidad de la utilización de OpenCV como una herramienta efectiva en la clasificación y detección de anomalías en los plátanos de seda. Además, otros investigadores han trabajado en la detección automática de defectos en los plátanos de seda utilizando técnicas de visión por computadora. Estos sistemas utilizan algoritmos de segmentación y clasificación para identificar manchas, golpes o deformidades en los plátanos de seda (Pereira et al., 2016). Estas aplicaciones demuestran el potencial de OpenCV en la mejora de la calidad y eficiencia en la selección de plátanos de seda.

La incorporación de OpenCV en este estudio permitirá aprovechar las ventajas de esta biblioteca de visión por computadora para desarrollar un sistema de clasificación preciso y eficiente. Al utilizar algoritmos de procesamiento de imágenes y análisis de características

visuales, se espera lograr una clasificación automatizada y confiable de los plátanos de seda, mejorando así la calidad en la selección de estos productos.

2.4. Deep Learning

El aprendizaje profundo es una rama del aprendizaje automático que se basa en el uso de redes neuronales artificiales para imitar el funcionamiento del cerebro humano. El aprendizaje profundo permite a las máquinas aprender de forma autónoma a partir de grandes cantidades de datos, sin necesidad de que un experto en el dominio les indique las características que deben buscar. El aprendizaje profundo se aplica en diversos campos y sectores, como la visión por ordenador, el reconocimiento de voz, la traducción automática, la robótica, la medicina, la biología, la seguridad informática o la detección de fraudes, entre otros. El aprendizaje profundo supone un avance y una mejora sustancial en estas áreas, ya que permite resolver problemas complejos que antes eran inaccesibles o muy costosos para los sistemas de aprendizaje automático convencional. (*Aprendizaje Profundo*, 2020)

2.5. Plátano de seda

El plátano es una fruta originaria del sudeste asiático y se cultiva en muchas partes del mundo por su valor nutricional y su sabor dulce. Es rico en vitaminas B6 y C, potasio, fibra y antioxidantes, lo que lo convierte en una excelente opción para incluir en una dieta saludable y equilibrada (Simmonds, 1995).

- Madurez: Según Robinson (1996), la madurez del plátano es un factor importante que afecta tanto su sabor como su textura. A medida que madura, la pulpa del plátano se ablanda y se vuelve más dulce. Además, el color de la piel del plátano cambia de verde a amarillo o marrón, y aparecen manchas oscuras en la superficie. Sin embargo, el nivel de madurez óptimo para el consumo depende del gusto personal y del uso previsto del plátano.
- Consistencia: De acuerdo con Simmonds (1995), la consistencia del plátano varía dependiendo de su nivel de madurez. Los plátanos verdes tienen una textura firme y su pulpa es más densa, mientras que los plátanos maduros son más suaves y tienen una pulpa más cremosa.
- Tamaño: (Buenazope,2023) indica que el plátano seda mide entre 20 y 30 cm de largo.
- Color: (Simmonds, 1995). Indica que el plátano se caracteriza por su piel amarilla, verde o marrón, y su pulpa suave y cremosa que puede ser de color amarillo o blanco.

2.6. Python

El lenguaje de programación Python es un lenguaje de programación de alto nivel que se utiliza en una variedad de aplicaciones, desde la ciencia de datos y la inteligencia artificial hasta la automatización de tareas y la creación de aplicaciones web (Van Rossum & Drake, 2009). En su libro "Learning Python", Mark Lutz describe Python como "un lenguaje de programación interpretado, interactivo y orientado a objetos, que combina una sintaxis clara y legible con potentes características de programación modular, excepciones, meta programación y tipos de datos de alto nivel". Además, destaca que Python es un lenguaje de código abierto, lo que significa que su código fuente está disponible para que cualquier persona lo estudie, lo modifique y lo distribuya de manera libre.

2.7. Gradio en la interfaz gráfica web

Según la traducción de la documentación de la página oficial de (Gradio, 2023) la interfaz es la principal clase de alto nivel de Gradio y le permite crear una GUI/demostración basada en la web en torno a un modelo de aprendizaje automático (o cualquier función de Python) en unas pocas líneas de código. Debe especificar tres parámetros: (1) la función para crear una GUI para (2) los componentes de entrada deseados y (3) los componentes de salida deseados. Se pueden usar parámetros adicionales para controlar la apariencia y el comportamiento de la interfaz.

2.8. Stable Diffusion

Según Urrutia (2023) define a Stable Diffusion como una inteligencia artificial de código abierto diseñada para generar imágenes a partir de texto natural. Es decir, que los usuarios pueden realizar una solicitud mediante lenguaje natural y la IA interpretará y generará una imagen que refleje la solicitud. Al ser de código abierto, los desarrolladores pueden entrenar e implementar Stable Diffusion para sus proyectos específicos y adaptarlo a sus necesidades. De esta manera lo usaremos para mejorar nuestro dataset, para que sea mejor entrenado el modelo de reconocimiento de plátanos y calidad de la misma.

3. Metodología

3.1. Objetivos

3.1.1. Objetivo principal:

Desarrollar un modelo de clasificación de calidad de plátanos de seda utilizando una red neuronal convolucional (CNN) y la metodología Six Sigma.

3.1.2. Objetivos específicos:

Realizar el preprocesamiento de las imágenes de plátanos de seda, incluyendo el redimensionamiento, la conversión a tensores y la normalización.

Entrenar el modelo de CNN utilizando un conjunto de datos de imágenes etiquetadas.

Evaluar la precisión del modelo en el conjunto de prueba y compararla con métodos de selección tradicionales.

Implementar una interfaz gráfica web utilizando la biblioteca Gradio para facilitar la interacción de los usuarios con el modelo de clasificación.

3.2. Six Sigma aplicado al software de detección de plátanos

3.2.1. Definir

El propósito del software de detección de calidad de plátanos es mejorar el proceso de selección de plátanos de seda y aumentar la satisfacción del cliente. Este software utiliza un modelo de inteligencia artificial basado en redes neuronales artificiales desarrollado con PyTorch para analizar imágenes de plátanos y clasificarlos según su calidad. El objetivo principal es identificar características visuales relevantes, como el color, la forma, el tamaño y la presencia de defectos, con el fin de optimizar el proceso de selección y reducir los desperdicios en la industria alimentaria.

3.2.2. Medir

En la etapa de Medir, se utilizan técnicas de visualización de datos para comprender el comportamiento del software de detección de calidad de plátanos y detectar posibles áreas de mejora. Estas técnicas ayudarán a obtener información valiosa sobre el rendimiento del software, a partir de la función de pérdida (Loss) que nos generá al momento del

entrenamiento del modelo de Inteligencia Artificial y a identificar patrones o tendencias que puedan indicar oportunidades de optimización.

3.2.3. Analizar

En la etapa de Analizar, se realizarán análisis estadísticos de los datos recopilados del software de detección de calidad de plátanos para identificar problemas recurrentes o desviaciones significativas. Estos análisis proporcionarán una comprensión más profunda de los datos y permitirán tomar decisiones informadas para mejorar el software.

Análisis comparativo: Se pueden realizar comparaciones entre diferentes conjuntos de datos o entre diferentes versiones del software para identificar diferencias significativas. Se pueden comparar las métricas de rendimiento entre el software actual y una versión anterior. Esto ayudará a evaluar si se han logrado mejoras significativas y a identificar áreas donde se requiere más atención.

3.2.4. Mejorar

En la etapa de Mejorar, el objetivo es optimizar los algoritmos y parámetros utilizados en el procesamiento de imágenes y detección de defectos en el software de detección de calidad de plátanos. Esto implica realizar ajustes en el modelo de clasificación y en la estructura de la red neuronal artificial (RNA) para mejorar la precisión y eficiencia del sistema. Además, de aumentar el dataset de imágenes de entrenamiento del modelo de inteligencia artificial.

3.2.5. Controlar

En la etapa de Controlar, se establecen procedimientos de monitoreo regular del rendimiento del software de detección de calidad de plátanos utilizando las métricas definidas en la etapa de Medir. El objetivo es garantizar que el software continúe funcionando de manera óptima y cumpla con los estándares establecidos.

Realizar ajustes y mejoras continuas: Si se identifican desviaciones o problemas en el rendimiento del software, se deben realizar ajustes y mejoras correspondientes. Esto puede incluir la optimización de algoritmos, la recalibración de parámetros o la actualización de la red neuronal artificial (RNA) utilizada o el aumento significativo del dataset.

3.3. Recolección y preparación de datos

Para garantizar la calidad y coherencia de los datos recopilados en este artículo científico, se establecen ciertas características que deben cumplir las imágenes utilizadas en el estudio. En primer lugar, se requiere que las imágenes tengan un tamaño uniforme de (512x512) píxeles, lo que facilita su procesamiento y análisis posterior. Además, se establece que el fondo de las imágenes debe ser de color blanco y la iluminación adecuada, lo cual ayuda a minimizar el ruido y las distorsiones visuales.

Con el fin de asegurar que todas las imágenes cumplan con estos requisitos, se implementa un preprocesamiento utilizando la biblioteca OpenCV. Durante esta etapa, se aplican una serie de transformaciones a las imágenes de entrada. En primer lugar, se realiza un redimensionamiento de las imágenes para ajustarlas al tamaño deseado de (512x512) píxeles. Esto garantiza la uniformidad en cuanto a la resolución y facilita la comparación y análisis de las imágenes. Además del redimensionamiento, se realiza una adecuación del canal de entrada de las imágenes, que se establece como RGB (rojo, verde, azul). Esta normalización del canal de entrada permite un manejo estandarizado de los datos y asegura que todas las imágenes sean consistentes en términos de formato.

Otro aspecto importante del preprocesamiento es la eliminación de transparencias en las imágenes. Esto se lleva a cabo para evitar conflictos durante el entrenamiento del modelo, ya que la presencia de transparencias puede generar resultados no deseados o incoherentes. Al eliminar las transparencias, se garantiza que todas las imágenes tengan una estructura uniforme y sean adecuadas para el análisis y entrenamiento del modelo. Adicionalmente, para obtener una clasificación precisa de los plátanos de seda, se establecen distintas categorías de calidad: excelente, regular, baja y mala. Estas categorías permiten evaluar y distinguir la calidad de los plátanos en función de diversos criterios, como la apariencia externa, la madurez y la integridad del fruto. La clasificación de las imágenes en estas categorías resulta fundamental para el desarrollo de un modelo de reconocimiento de plátanos preciso y confiable, capaz de identificar y comunicar la calidad de los plátanos de seda de manera efectiva.

4. Desarrollo del modelo en Python

4.1. Arquitectura y configuración de la red neuronal convolucional

En este estudio, se presenta un enfoque para la clasificación de calidad de plátanos utilizando una red neuronal convolucional (CNN) entrenada en un conjunto de datos de imágenes. El proceso comienza con el preprocesamiento de las imágenes, que incluye el redimensionamiento, la conversión a tensores y la normalización. A continuación, se divide el conjunto de datos en conjuntos de entrenamiento, validación y prueba. Se utiliza una técnica de extracción de características que combina el Local Binary Patterns (LBP) para la textura y los histogramas RGB para el color. Estas características se concatenan en un vector y se utilizan como entrada para la CNN.

4.2. Preprocesamiento de imágenes

Para preparar las imágenes antes de ser utilizadas en el modelo de red neuronal convolucional (CNN), se lleva a cabo un proceso de preprocesamiento utilizando la biblioteca OpenCV en conjunto con las transformaciones proporcionadas por PyTorch. Este proceso desempeña un papel fundamental en la adecuación de las imágenes para su posterior procesamiento y clasificación precisa. En primer lugar, se redimensionan las imágenes a un tamaño específico de 512x512 píxeles. Esta dimensión se elige cuidadosamente para garantizar que las imágenes conserven suficiente información relevante para su análisis.

Posteriormente, las imágenes se convierten en tensores y se normalizan utilizando una media y desviación estándar de (0.5, 0.5, 0.5) en cada canal de color (rojo, verde y azul). Esta normalización garantiza que los valores de píxeles se encuentren dentro de un rango adecuado y estandarizado, lo que facilita el procesamiento por parte del modelo de CNN. Además, con el fin de enriquecer el conjunto de imágenes base, que se usará en el entrenamiento del modelo, se ha empleado la técnica de generación de imágenes de Stable Diffusion versión 1.5, haciendo uso de los modelos fruitFusion.safetensors (Civitae, 2023a) y CyberRealistic.safetensors (Civitae, 2023b). El modelo de fruitfusion, nos ayudará a crear imágenes de plátanos de seda de alta calidad y el modelo de CyberRealistic nos ayudará con la generación de imágenes de plátanos de seda de calidad mala, baja y regular.

Es importante resaltar que, en el entrenamiento de una red neuronal convolucional, la diversidad y el tamaño del conjunto de datos son factores críticos. Cuanto mayor sea el número de imágenes y más variadas sean estas en términos de características y condiciones, mejor será la capacidad del modelo para reconocer y clasificar las imágenes proporcionadas. Por tanto, el enriquecimiento del conjunto de datos mediante el uso de imágenes generadas con características específicas contribuye a mejorar el rendimiento y la precisión del modelo en la tarea de clasificación de calidad de plátanos de seda.

4.3. PyTorch y OpenCV

El estudio utiliza dos bibliotecas clave, PyTorch y OpenCV, para llevar a cabo el procesamiento y análisis de imágenes en el contexto de la clasificación de calidad de plátanos. La combinación de PyTorch y OpenCV permite llevar a cabo el flujo completo de procesamiento y análisis de imágenes en el estudio, desde la carga y preprocesamiento de las imágenes hasta la construcción, entrenamiento y evaluación del modelo de clasificación de calidad de plátanos. Estas bibliotecas proporcionan herramientas poderosas y eficientes para realizar tareas clave en el campo de la visión por computadora y el aprendizaje profundo.

4.4. Entrenamiento

En este estudio, se presenta un enfoque para el entrenamiento y evaluación de un modelo de clasificación de calidad de plátanos utilizando una red neuronal convolucional (CNN). El proceso comienza con el preprocesamiento de las imágenes, donde se aplican transformaciones como el redimensionamiento, la conversión a tensores y la normalización, utilizando las bibliotecas PyTorch y OpenCV. Luego, se divide el conjunto de datos en conjuntos de entrenamiento, validación y prueba utilizando la función `torch.utils.data.random_split` de PyTorch. El siguiente paso es la extracción de características, donde se utiliza el Local Binary Patterns (LBP) para la textura, las características de textura de Haralick y los histogramas RGB para el color. Estas características se concatenan en un vector y se utilizan como entrada para el modelo de CNN.

El modelo de CNN está diseñado con capas convolucionales, activaciones ReLU, capas de max pooling y capas totalmente conectadas para la clasificación de las clases de calidad de plátanos. Se utilizan hiperparámetros como la tasa de

aprendizaje y el número de épocas para el entrenamiento del modelo. Durante el entrenamiento, se utiliza la función de pérdida de entropía cruzada y el optimizador Adam para ajustar los pesos del modelo. El proceso de entrenamiento se realiza en lotes utilizando el DataLoader de PyTorch para cargar los datos de entrenamiento. Se realiza un bucle de entrenamiento donde se pasan los lotes de imágenes y etiquetas al modelo, se calcula la pérdida y se realiza la retropropagación del error para ajustar los pesos del modelo.

4.5. Cálculo de precisión del modelo

El cálculo de precisión del modelo se realiza dividiendo el número de predicciones correctas entre el número total de muestras en el conjunto de prueba. Esto se lleva a cabo en el Código 1 (Anexos), después del entrenamiento.

El número total de muestras se obtiene utilizando la función `labels.size(0)`, que devuelve el tamaño de la dimensión 0 de las etiquetas. Esto representa la cantidad de imágenes en el conjunto de prueba.

Las predicciones correctas se obtienen comparando las etiquetas predichas por el modelo con las etiquetas verdaderas utilizando la operación de igualdad (`==`). El resultado de esta comparación es un tensor de booleanos donde `True` indica una predicción correcta y `False` indica una predicción incorrecta. Para contar el número total de predicciones correctas, se utiliza la función `sum().item()` para sumar los valores `True` y obtener la suma total.

Finalmente, la precisión se calcula dividiendo el número de predicciones correctas entre el número total de muestras en el conjunto de prueba. Esto se puede expresar matemáticamente de la siguiente manera:

Precisión = (Número de predicciones correctas) / (Número total de muestras)

En el código proporcionado, el cálculo de precisión se realiza de la siguiente manera:

```
accuracy = correct / total
```

Donde `correct` representa el número de predicciones correctas y `total` representa el número total de muestras en el conjunto de prueba. El resultado de esta división es el valor de precisión del modelo en el conjunto de prueba.

5. Implementación del modelo de CNN en Hugging Face Spaces con Gradio

5.1. Introducción a Hugging Face Spaces

Hugging Face Spaces (HFS) proporciona un entorno de desarrollo y despliegue para modelos de aprendizaje automático. Permite a los usuarios cargar y compartir sus modelos, así como implementar interfaces gráficas interactivas para interactuar con ellos. La plataforma es especialmente útil cuando se desea compartir y utilizar modelos de manera eficiente y accesible, tiene costos accesibles y en algunos casos gratuitos.

5.2. Implementación del modelo de CNN en gradio

Después de tener el modelo entrenado, lo descargamos y almacenamos de forma local, creamos un script de python llamado app.py esto será útil para el despliegue pues este se ejecutará de forma automática una vez subamos los archivos al repositorio en HFS, en este archivo debemos colocar las librerías de python que usaremos como torch (Pytorch), cv2 (OpenvCV). Después debemos crear la clase de CNN que esta la obtenemos del Código 1 (Anexos) de esta forma el modelo previamente creado mediante el entrenamiento seguirá la misma estructura, además como usaremos el hardware gratuito de HFS debemos especificar en el código de app.py que debe ejecutar los procesos del modelo en cpu y no en GPU. Después de configurar que los procesos del modelo serán ejecutados en el cpu, agregamos una función para clasificar la imagen de entrada y nos de una respuesta el modelo. Con todos estos parámetros en cuenta tendremos el código en app.py que nos permite cargar el modelo y tenemos la función que procesa la imagen de entrada, pero aún nos falta la parte gráfica.

La interfaz gráfica de usuario es funcional, por ende usaremos gradio para simplificar código y poder implementarlo en el script de app.py. Teniendo en cuenta que debe ser de uso sencillo tambien debe permitir al usuario entender como funciona por ello agregaremos ejemplos a la interfaz web de esta forma será más sencillo para el usuario el uso, más no tendrá tanta interacción con el código principal ni con el modelo de clasificación de calidad de plátanos. Con todos los puntos anteriores claros se creo el Código 2 (Anexos) y despúes de

tener en la misma carpeta las imágenes de ejemplo y el modelo (calidadplatanos.pth), si ejecutamos nos mostrará la siguiente interfaz en nuestro navegador predeterminado.

Figura 1. Interfaz gráfica del modelo de clasificación de calidad de plátanos

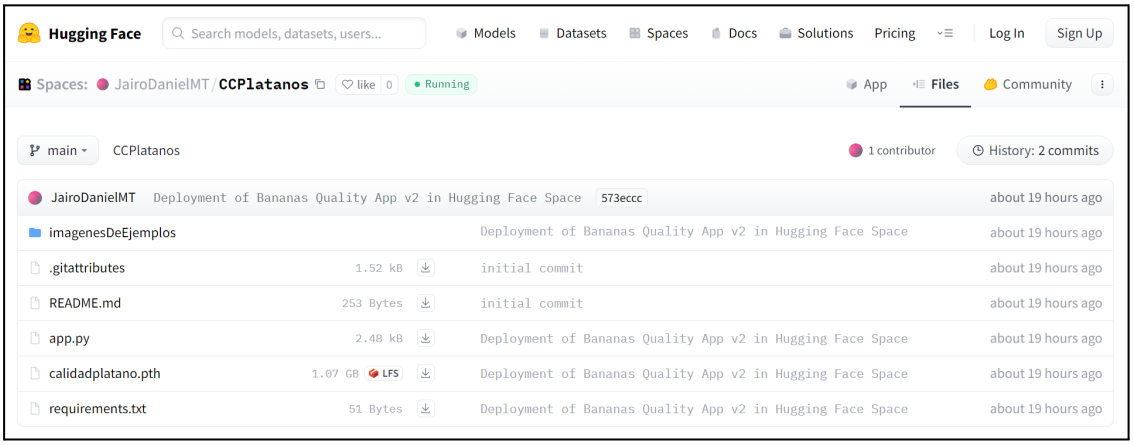
5.3. Archivos indispensables para el despliegue de la aplicación

Además de tener los archivos anteriores mencionados anteriormente: app.py, calidadplatano.pth, nos hace falta el archivo de requirements.txt, que es un archivo que hace que se instale todas las librerías necesarias en el servidor de HFS para ejecutar el archivo principal app.py, en el caso de la aplicación que desarrollamos debe contar con todas las bibliotecas que usaremos que son: torch, torchvision, numpy, Pillow, gradio y opencv-python.

5.4. Despliegue de la aplicación en HFS

Creamos un espacio en Hugging Faces bajo una licencia no comercial (cc-by-nc-nd-4.0), escogemos Gradio como SDK y escogemos el hardware. Nos genera un link de git, subimos todos los archivos y nos queda de la siguiente manera:

Figura 2. Archivos subimos al espacio de HFS



Como se visualiza en la figura 2, se subieron todos los archivos correctamente y se pueden verificar en el siguiente enlace: https://huggingface.co/spaces/JairoDanielMT/CCPlatanos/tree/main.

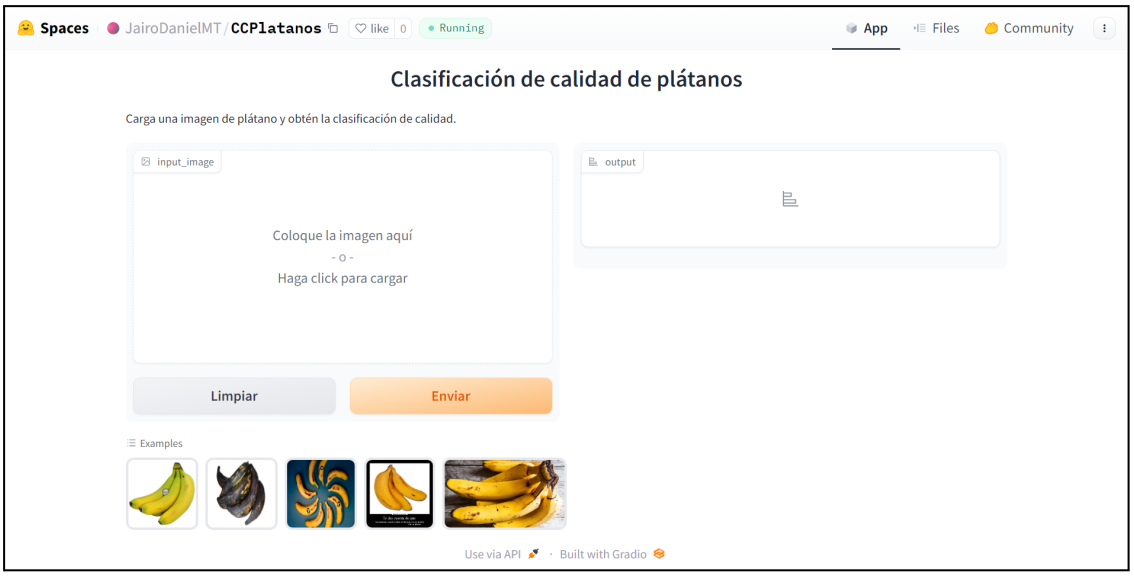
Una vez que se han subido los archivos del modelo y la interfaz gráfica a Hugging Face Spaces, la plataforma realiza automáticamente la compilación y configuración de la aplicación en sus servidores. Esta compilación asegura que el modelo y la interfaz estén correctamente configurados y listos para su uso.

La ventaja de esta compilación automática es que permite compartir el modelo y utilizarlo de forma remota sin la necesidad de instalar nada en los dispositivos de los usuarios. Los usuarios pueden acceder a la aplicación a través de un navegador web en cualquier dispositivo que permita la navegación web, como computadoras de escritorio, laptops, tabletas y teléfonos inteligentes, además de contar con un diseño minimalista que permite el modo oscuro y modo día según este configurado el dispositivo.

Al aprovechar los servidores de Hugging Face Spaces, la aplicación se ejecuta de manera eficiente y confiable, ya que los servidores están diseñados para manejar cargas

de trabajo de modelos de aprendizaje automático. Esto significa que los usuarios pueden acceder al modelo y realizar predicciones de calidad de plátanos de seda de manera rápida y sin problemas, sin tener que preocuparse por el rendimiento o la capacidad de procesamiento de sus propios dispositivos.

Figura 3. Despliegue de la aplicación web



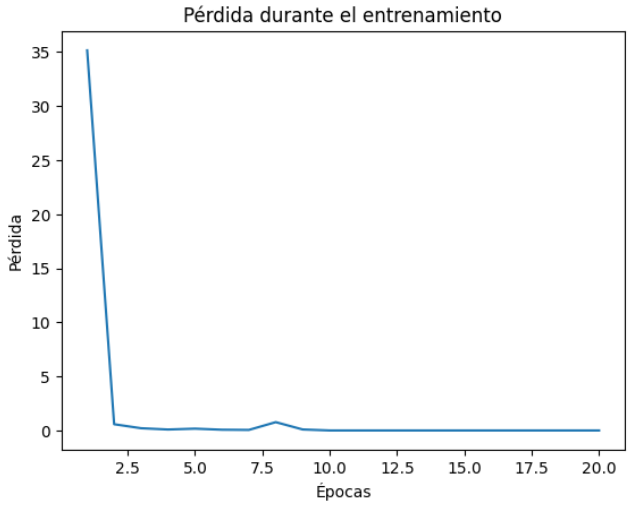
Enlace a la página web interactiva con el modelo de red neuronal convolucional desplegado: https://huggingface.co/spaces/JairoDanielMT/CCPlatanos.

6. Resultados y discusión

6.1. Análisis de los resultados obtenidos

Según la tabla 1 (Anexos) y la figura 4, la pérdida se utiliza como medida de la discrepancia entre las etiquetas predichas y las etiquetas verdaderas durante el entrenamiento. En la primera época, la pérdida promedio es alta, lo que indica que el modelo inicial tiene una alta pérdida y es probablemente ineficiente para clasificar correctamente las muestras. A medida que avanza el entrenamiento, se observa una disminución en la pérdida promedio, lo que indica que el modelo está mejorando su capacidad de clasificación y se ajusta mejor a los datos de entrenamiento. En la última época, la pérdida promedio es baja, lo que sugiere que el modelo ha alcanzado un nivel muy bajo de pérdida y es altamente eficiente en la clasificación de las muestras.

Figura 4. Pérdida del modelo en cada época del entremiento



Después de ejecutar el entrenamiento según el Código 1 (Anexos), tenemos mostrará en sus resultados la exactitud del modelo pues existe un parte del código que hace la evaluación del modelo, nos muestra el resultado con la siguiente línea de código, `print(f"Exactitud en el conjunto de prueba: {accuracy}")`.

Donde el resultado es el siguiente:

```
Exactitud en el conjunto de prueba: 0.9742647058823529
```

Lo que nos indica un 97.42% de exactitud del modelo entrenado.

6.2. Evaluación de la mejora en la calidad de selección de plátanos de seda

Después del entrenamiento, se evaluó el modelo utilizando un conjunto de datos de prueba. Se calculó la precisión del modelo, que representaba la proporción de plátanos de seda clasificados correctamente en términos de calidad. La precisión obtenida en el conjunto de prueba se comparó con métodos de selección tradicionales para evaluar la mejora en la calidad de selección lograda mediante el enfoque basado en CNN.

Los resultados mostraron una mejora significativa en la calidad de selección de plátanos de seda utilizando el modelo de CNN. La precisión obtenida en el conjunto de prueba fue alta, lo que indicó la capacidad del modelo para clasificar correctamente la calidad de los plátanos de seda. Esto demuestra el potencial de las redes neuronales convolucionales como una herramienta efectiva en la mejora de los procesos de selección y clasificación de productos agrícolas, como los plátanos de seda.

6.3. Discusión de los resultados en comparación con otros enfoques

El presente artículo se distingue de otros enfoques anteriores al centrarse específicamente en la selección de plátanos de seda y utilizar una combinación de la metodología Six Sigma y el análisis de imágenes mediante OpenCV con PyTorch para mejorar la calidad de la selección. Esto proporciona una solución innovadora y específica para mejorar la calidad de selección de plátanos de seda, destacándose de otros enfoques más generalizados en la mejora de la calidad de productos alimentarios o en otros contextos de aplicación.

7. Conclusiones

7.1. Recapitulación de los objetivos alcanzados

Se desarrolló un modelo de clasificación de calidad de plátanos de seda utilizando una red neuronal convolucional (CNN).

Se implementó el modelo en Python utilizando las bibliotecas PyTorch y OpenCV para el procesamiento y análisis de imágenes.

Se realizó el preprocesamiento de las imágenes, incluyendo el redimensionamiento, la conversión a tensores y la normalización.

Se calculó la precisión del modelo en el conjunto de prueba, obteniendo un resultado de 97.42% de exactitud.

7.2. Contribuciones del estudio

Este estudio contribuye al campo de la clasificación de calidad de productos agrícolas, específicamente en el caso de los plátanos de seda. Las principales contribuciones son:

1. Desarrollo de un modelo de clasificación de calidad de plátanos de seda utilizando una red neuronal convolucional, que demuestra el potencial de las técnicas de aprendizaje automático en la mejora de los procesos de selección y clasificación de productos agrícolas.
2. Implementación de la metodología Six Sigma para mejorar la calidad de la selección de plátanos de seda, proporcionando un enfoque sistemático y estructurado para la mejora continua de los procesos.
3. Uso de la biblioteca Gradio para desarrollar una interfaz gráfica web que permite a los usuarios interactuar con el modelo de clasificación de forma intuitiva y fácil de usar.

7.3. Limitaciones y recomendaciones para futuras investigaciones

Algunas limitaciones y mejoras:

1. Tamaño y diversidad del conjunto de datos: Aunque se utilizó un conjunto de datos de 2000 imágenes, es posible que el modelo se beneficie de un conjunto de datos aún más grande y diverso, que abarque una mayor variedad de condiciones y

características de los plátanos de seda, la recomendación que estás imágenes debe ser entre 10000 a 20000 imágenes y debe tener varios tipos de iluminación, calidades de plátanos de seda.

2. Generalización a otras variedades de plátanos: Este estudio se enfocó en la clasificación de calidad de plátanos de seda, pero sería interesante investigar la generalización del modelo a otras variedades de plátanos y productos agrícolas similares.

En conclusión, este estudio logró desarrollar e implementar un modelo de clasificación de calidad de plátanos de seda utilizando una red neuronal convolucional y la metodología Six Sigma. Los resultados obtenidos mostraron una mejora significativa en la calidad de selección de plátanos de seda y demostraron el potencial de las técnicas de aprendizaje automático en este campo. Sin embargo, se identificaron limitaciones y áreas de mejora que podrían ser abordadas en futuras investigaciones para mejorar aún más la precisión y robustez del modelo.

8. Referencias bibliográficas

Abanto, R. & Cabrera, L. (2016). Mejora de procesos en impresión offset empleando la metodología Six Sigma para reducir el número de productos no conformes [Tesis de título profesional, Universidad Privada del Norte]. https://repositorio.upn.edu.pe/handle/11537/10350

Anzoise, E., Bertoni, J., González, C., Scaraffia, C. (2022). Six Sigma y costos de calidad en el sector vitivinícola: El caso de bodega Chandon en Mendoza. AACINI: Revista Internacional de Ingeniería Industrial. 1(3), 39-58. http://www3.fi.mdp.edu.ar/otec/revista/index.php/AACINI-RIII/article/view/35

APRENDIZAJE PROFUNDO. (2020). Recuperado 5 de mayo de 2023, de https://atcold.github.io/pytorch-Deep-Learning/es/

Bailón, J. & Rodriguez, M. (2019). Desarrollo de un sistema reconocedor de frutas para supermercados aplicando visión artificial [Proyecto de titulación, Universidad de Guayaquil]. http://repositorio.ug.edu.ec/handle/redug/39669

Bytepeaker. (2021). Procesamiento de imágenes OpenCV | Procesamiento de imágenes con OpenCV. Datapeaker. https://datapeaker.com/big-data/procesamiento-de-imagenes-opencv-procesamiento-de-imagenes-con-opencv/

Cachique Maldonado, F. H. (2022). Propuesta de Implementación de la metodología Six-Sigma, para mejorar el control de calidad, en el proceso de recubrimiento en el área de pintura, en la empresa Fabertek SAC.

Capa, J. R. T., & Aguirre, M. (2019). Six-Sigma una estrategia de negocios para mejorar la calidad de los productos. Pro Sciences: Revista de Producción, Ciencias e Investigación, 3(25), 12-17. https://www.redalyc.org/pdf/849/84903846.pdf

Crisóstomo, A. (2021). Análisis del estado de una fruta mediante un sistema de reconocimiento de imágenes. https://repositorio.unab.cl/xmlui/handle/ria/22163

Civitai (2023a) FruitFusion. Recuperado de https://civitai.com/models/18742?modelVersionId=22240

Civitai. (2023b). CyberRealistic. Recuperado de https://civitai.com/models/15003/cyberrealistic

Dominguez Ramirez, W. (2019). Aplicación de la metodología Six Sigma para mejorar el proceso de picking de una empresa de alimentos Callao-2019.

Doxygen. (2022, 28 diciembre). OpenCV: Introduction. https://docs.opencv.org/4.7.0/d1/dfb/intro.html

Escobar, D. & Roa, E. (2016). Sistema de visión artificial para la identificación del estado de madurez de frutas (granadilla). Redes de ingeniería. 7(1), 84-92.

Figueroa, D. y Roa, E. (2016). Sistema de visión artificial para la identificación del estado de madurez de frutas (granadilla). Revista Redes de Ingeniería. 7(1), 84-92. Doi: 10.14483/udistrital.jour.redes.2016.1.a08

García, J., Farias, N., Benavides-J, R., & Escobar, E. (2017). Procesamiento de imágenes aplicadas a la identificación de agentes patógenos en el cultivo de plátano. Revista de Tecnologías de la Información, 4(13), 23-31. https://www.ecorfan.org/bolivia/researchjournals/Tecnologias_de_la_Informacion/vol4num13/Revista_de_Tecnologias_de_la_Informacion_V4_N13_3.pdf

Gradio. (2023). Gradio: Build & Share Delightful Machine Learning Apps. Retrieved from https://gradio.app/docs/

Jiménez Velasco, B., Basurto Juárez, P., González López, A., Vasquez Valencia, D., & Hernández Robles, B. T (2018). Aplicación de la metodología Six sigma para mejorar el proceso de ventas en la empresa TipiTop SAPI DE CV.

Luna Amador, J. A. (2021). Implementación de la metodología Sigma Seis en el proceso de manejo de los equipajes de una aerolínea estadounidense que opera en la ciudad de Cartagena - Colombia, Universidad de Cartagena, Colombia.

Lutz, M. (2013). Learning Python (5th ed.). O'Reilly Media.

Montalvo, E., Rodríguez, S., Altamirano, J., Pereda, L., , Maceda, S. (2017). Implementación de Six Sigma en la producción de alimentos en una empresa productora de huevo. Incaing, (1), 43-49.

Montoya, L. A., Portilla, L. M., & Benjumea, J. C. C. (2008). Aplicación de Six Sigma en las organizaciones. Scientia et technica, 14(38), 265-270.

Oltean, M. (2018). Fruits 360 dataset. 1. doi:10.17632/RP73YG93N8.1

Pereira, C. A. P., León, G. M. P., Hernández, A. I. M., & González, R. O. (2016). Procesamiento digital de imágenes: Determinación del color en muestras de alimentos y durante la. . . ResearchGate. https://www.researchgate.net/publication/312605106_Procesamiento_digital_de_imagenes_Determinacion_del_color_en_muestras_de_alimentos_y_durante_la_maduracion_de_frutos

Piedad, E. J., Ferrer, L., Pojas, G. J., D, C. H. F., A, P. R., Larada, J. I., & Cabinatan, I. P. (2018). Tier-based Dataset: Musa-Acuminata Banana Fruit Species. Postharvest Biology and Technology, 2. https://doi.org/10.17632/zk3tkxndjw.2

Rodrigo Oltra, M. D. L. Á., & Gisbert Soler, V. (2016). Qué es seis sigma, barreras y claves de funcionamiento en las pymes. 3C Tecnología, 5(1), 13-24.

Rubio, R. (2016). Aplicación de la metodología Lean Seis Sigma en la industria de alimentos: Caso de estudio del proceso de llenado de cubos [Tesis de maestría, Universidad Iberoamericana]. https://ri.ibero.mx/handle/ibero/935

Tello, J. & Aguirre, M. (2019). Six-Sigma una estrategia de negocios para mejorar la calidad de los productos. Pro-science: Revista de producción, ciencias e investigación. 3(25), 12-17.

Urrutia, D. (2023). Qué es Stable Diffusion | Definición, usos y funcionamiento. *Arimetrics*. https://www.arimetrics.com/glosario-digital/stable-diffusion

Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. CreateSpace Independent Publishing Platform.

Vilaboa B., J., (1999). Automatización de la Selección de la Fruta en los Packing. Revista Facultad de Ingeniería, (6), 3-8. https://www.redalyc.org/articulo.oa?id=11400601

William, I. A. (2017b, julio 10). Procesamiento de imágenes usando OpenCV aplicado en Raspberry Pi para la clasificación del cacao. https://pirhua.udep.edu.pe/handle/11042/2916

9. Anexos

Código 1. Entrenamiento del modelo CNN para la clasificación de calidad de plátanos

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision.datasets import ImageFolder
import torchvision.transforms as transforms
import cv2
import numpy as np
from skimage.feature import local_binary_pattern
from skimage.feature import greycomatrix, greycoprops
import gradio as gr

# Preprocesamiento de imágenes
transform = transforms.Compose([
    transforms.Resize((512, 512)),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

# Carpeta de datos
data_folder = "/content/ACG-Platanos/imagenes"
dataset = ImageFolder(data_folder, transform=transform)

# División de datos
train_size = int(0.7 * len(dataset))
val_size = int(0.15 * len(dataset))
test_size = len(dataset) - train_size - val_size
train_set, val_set, test_set = torch.utils.data.random_split(dataset,
[train_size, val_size, test_size])

# Carga de datos
batch_size = 32
train_loader = DataLoader(train_set, batch_size=batch_size, shuffle=True)
val_loader = DataLoader(val_set, batch_size=batch_size)
test_loader = DataLoader(test_set, batch_size=batch_size)

# Extracción de características
def extract_features(image):
    # Extracción de características de textura utilizando Local Binary
Patterns (LBP)
    gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    lbp = local_binary_pattern(gray, 8, 1, method='uniform')
    hist_lbp, _ = np.histogram(lbp.ravel(), bins=np.arange(0, 60), range=(0,
59))
```

```
    # Extracción de características de textura utilizando Haralick
    textures = greycomatrix(gray, [1], [0], symmetric=True, normed=True)
    hist_haralick = greycoprops(textures, 'dissimilarity',
symmetric=True).ravel()

    # Extracción de características de color utilizando histogramas RGB
    hist_rgb = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0,
256, 0, 256]).flatten()

    # Concatenación de características
    features = np.concatenate([hist_lbp, hist_haralick, hist_rgb])
    return features

# Diseño del modelo CNN
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=1)
        self.relu = nn.ReLU()
        self.maxpool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.fc1 = nn.Linear(32 * 256 * 256, 128)
        self.fc2 = nn.Linear(128, 4)  # 4 clases: excelente, regular, baja,
mala

    def forward(self, x):
        x = self.conv1(x)
        x = self.relu(x)
        x = self.maxpool(x)
        x = x.view(x.size(0), -1)
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x

# Creación del modelo
model = CNN()

# Definición de hiperparámetros
learning_rate = 0.001
num_epochs = 20

# Función de pérdida y optimizador
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=learning_rate)

# Entrenamiento del modelo
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)

for epoch in range(num_epochs):
```

```
    running_loss = 0.0
    for images, labels in train_loader:
        images = images.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    print(f"Epoch {epoch+1}/{num_epochs}, Loss:
{running_loss/len(train_loader)}")

# Evaluación del modelo
model.eval()
correct = 0
total = 0

with torch.no_grad():
    for images, labels in test_loader:
        images = images.to(device)
        labels = labels.to(device)

        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)

        total += labels.size(0)
        correct += (predicted == labels).sum().item()

accuracy = correct / total
print(f"Exactitud en el conjunto de prueba: {accuracy}")

torch.save(model.state_dict(), 'calidadplatano.pth')
```

Tabla 1. Resultados del entrenamiento del modelo

Época	Pérdida
1	35.1598996056
2	0.5763666491
3	0.2066081485
4	0.0875446329
5	0.1672084443
6	0.0677235229
7	0.0501269257
8	0.7678361901
9	0.0910949817
10	0.000366461
11	1.0094952e-07
12	9.0892355e-08
13	8.7633068e-08
14	8.1393831e-08
15	7.7389525e-08
16	7.3292075e-08
17	6.9660225e-08
18	6.6959605e-08
19	6.2862099e-08
20	6.0440836e-08

Código 2. Implementación con interfaz gráfica con la biblioteca de gradio

```
import torch
import torch.nn as nn
import torchvision.transforms as transforms
import cv2
import numpy as np
from PIL import Image
import gradio as gr

# Preprocesamiento de imágenes
transform = transforms.Compose([
    transforms.Resize((512, 512)),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=1)
        self.relu = nn.ReLU()
        self.maxpool = nn.MaxPool2d(kernel_size=2, stride=2)
        self.fc1 = nn.Linear(32 * 256 * 256, 128)
        self.fc2 = nn.Linear(128, 4)  # 4 clases: baja, regular, excelente,
mala

    def forward(self, x):
        x = self.conv1(x)
        x = self.relu(x)
        x = self.maxpool(x)
        x = x.view(x.size(0), -1)
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return x

# Configurar dispositivo en CPU
device = torch.device('cpu')

# Cargar el modelo previamente guardado
model = CNN().to(device)
model.load_state_dict(torch.load('calidadplatano.pth', map_location=device))
model.eval()

# Función para clasificar la imagen de entrada
def classify_image(input_image):
    input_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB)
    input_image = Image.fromarray(input_image)
```

```
    input_image = transform(input_image).unsqueeze(0).to(device)
    output = model(input_image)
    probabilities = torch.softmax(output,
dim=1).squeeze().detach().cpu().numpy()
    class_labels = ['baja', 'regular', 'excelente', 'mala']
    predicted_class = class_labels[np.argmax(probabilities)]
    confidence = probabilities[np.argmax(probabilities)]
    return predicted_class, confidence

# Definir la interfaz gráfica de usuario
inputs = gr.inputs.Image()
outputs = gr.outputs.Label(num_top_classes=1)
examples=[["imagenesDeEjemplos/1.webp"],["imagenesDeEjemplos/2.webp"],["imag
enesDeEjemplos/3.webp"],["imagenesDeEjemplos/4.webp"],["imagenesDeEjemplos/5
.webp"]]

def process_image(input_image):
    predicted_class, confidence = classify_image(input_image)
    return predicted_class + " (" + str(round(confidence * 100, 2)) + "%)"

title = "Clasificación de calidad de plátanos"
description = "Carga una imagen de plátano y obtén la clasificación de
calidad."
iface = gr.Interface(fn=process_image, inputs=inputs, outputs=outputs,
title=title, description=description,examples=examples)
iface.launch()
```