ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UNA RED NEURONAL CONVOLUCIONAL PARA LA CLASIFICACIÓN DE GRANOS DE CACAO CON PULPA

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

KEVIN PAÚL PROAÑO JARAMILLO

kevin.proano@epn.edu.ec

DIRECTOR: Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO

felipe.grijalva@epn.edu.ec

DMQ, octubre 2022

CERTIFICACIONES

Yo, KEVIN PAÚL PROAÑO JARAMILLO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

KEVIN PAÚL PROAÑO JARAMILLO

Certifico que el presente trabajo de integración curricular fue desarrollado por KEVIN PAÚL PROAÑO JARAMILLO, bajo mi supervisión.

Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO
DIRECTOR DEL TRABAJO DE INTEGRACIÓN

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

KEVIN PAÚL PROAÑO JARAMILLO

Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO

DEDICATORIA

Este trabajo esta dedicado a quienes han estado para mi en los días y noches mas difíciles y me han apoyado física, mental y emocionalmente: Wilma y Paola , quienes han sido mi apoyo incondicional.

A mi familia por ser un apoyo para mi, se que siempre cuento con ustedes para todo.

A mi ángel en el cielo que siempre nos protege y nos cuida.

Finalmente a mi, por no darme por vencido.

Paúl Proaño

AGRADECIMIENTO

Páginas y palabras faltarían para agradecer a mi madre y mi novia, quienes junto a Gody, Molly y Bonny, me han enseñado que, con amor, esfuerzo, constancia y perseverancia todo es posible; han sido mi apoyo incondicional y el motor de mi vida; ayudándome y apoyándome en los caminos por los que transito , todas mis metas son por y para ustedes mis 5 amores.

Resulta importante reconocer y agradecer el amor, orgullo y comprensión que mi padre y mi madre tuvieron hacia mi a lo largo, ya que fueron el pilar fundamental que trazo el camino, por el que me encuentro ahora aquí.

A mis 4 abuelitos, quienes con mucho cariño me han inculcado valores desde pequeño, y me han llenado de sabiduría, experiencia, amor y alegría la vida.

A Marquito que más que abuelo ha sido para mi, un poco padre, poco amigo y mi cómplice, gracias por todo y espero ser como tú en el futuro. Y a Zoilita que aunque físicamente no nos acompañe más, estuvo y estará para apoyarme desde donde se encuentre.

Un agradecimiento al PhD. Felipe Grijalva por toda la colaboración, paciencia y ayuda en la realización de este trabajo, así como por su forma única de transmitir conocimiento desde que me encontraba en las aulas de la Escuela Politécnica Nacional; institución a la que agradezco por todos los conocimientos que obtuve dentro y fuera de las aulas.

A mis amigos y compañeros, gracias por enseñarme que hay muchas más cosas que solo llegar a estudiar e irse en una universidad, su apoyo fue muy importante dentro y fuera de la carrera.

Gracias.

Paúl Proaño

ÍNDICE DE CONTENIDO

RESUMEN

El complejo proceso al que se ve sometido la clasificación de la pulpa de cacao en el Ecuador es, sin lugar a duda, un determinante cuando se quiere comprender el déficit que ha venido sufriendo la riqueza del grano ecuatoriano. Debido a que a lo largo del tiempo se ha realizado un proceso de clasificación en base a las propiedades organolépticas teniendo como indicadores tradicionales a la textura, el aroma, el sabor y su color. Es por ello que, dada la necesidad de ofrecer un tratamiento óptimo a los granos de cacao y, en beneficio de los cultivadores, grandes industrias, pequeños productores y consumidores, se ha propuesto la implementación de una Red Neuronal Convolucional (CNN) que facilite y agilice la clasificación de los granos de cacao con pulpa. Esta transferencia de aprendizaje, a través del CNN permitirá desarrollar un sistema capaz de reducir la tasa de error a la hora de clasificar los granos de cacao; así como disparar la calidad del grano “Nacional” del Ecuador caracterizado por su exquisito sabor y aroma.

PALABRAS CLAVE: Redes Neuronales Convolucionales, Pulpa de cacao, Transferencia de aprendizaje, clasificación, Ecuador, Aumento de datos.

ABSTRACT

The complex process to which the classification of cocoa pulp in Ecuador is subjected is undoubtedly a determining factor when it comes to understanding the deficit of the cocoa industry. The complex process to which the classification of cocoa pulp in Ecuador is subjected is undoubtedly a determining factor when it comes to understanding the deficit that the Ecuadorian bean has the deficit that the richness of the Ecuadorian bean has been suffering. Because over time, a classification process has been a process of classification based on organoleptic properties, with texture, texture, flavor, and flavor as traditional indicators. traditional indicators such as texture, aroma, flavor and color. That is why, the need to offer an optimal treatment to cocoa beans and, for the benefit of the growers, large industries the benefit of growers, large industries, small producers and consumers, the implementation of a Neuron Network has been proposed. the implementation of a Convolutional Neural Network (CNN) that facilitates and speeds up the classification of cocoa beans with the classification of cocoa beans with pulp. This transfer of learning, through the CNN the CNN will make it possible to develop a system capable of reducing the error rate when classifying cocoa beans. the quality of Ecuador's "Nationalçocoa bean, which is characterized by its exquisite flavor characterized by its exquisite flavor and aroma

KEYWORDS: Convolutional Neural Networks, Cocoa pulp, Transfer learning, Classification, Ecuador, Data Augmentation.

1. INTRODUCCIÓN

En el Ecuador, la clasificación de los granos de cacao supone un desafío de importancia para la industria cacaotera y todos quienes la conforman. El grano de cacao es uno de las materias primas con mas relevancia que tiene el país, ya que cuenta con varios viveros y cultivos a lo largo de sus territorios [1]: lo que lo sitúa como uno de los recursos que mas se exportan a nivel nacional e internacional [1], debido a que muchos agricultores de cacao se encuentran en Bolivia, Ecuador, Colombia y Costa Rica; representando el 70 % aproximadamente de la exportación de cacao. La materia prima del chocolate, el cacao, ha dejado de ser percibido en la sociedad como un dulce, un postre o una simple golosina; las nuevas maneras de ver al mundo han permitido estudiar al grano “Nacional” y resaltar sus características nutritivas y beneficiosas para la salud, no solo de los ecuatorianos sino también de quienes habitan en los territorios a los que se exporta este recurso. Al ser rico en antioxidantes ayuda a prevenir enfermedades cardiovasculares [2] y permite cuidar de nuestra salud. Es por ello que, su correcta clasificación, debería ser el paso mas importante.

La World Cocoa Foundation nos indica que el cacao ha tenido un crecimiento porcentual casi fijo del 3 % anual [1], por lo que el déficit de cacao aumenta cada temporada. Esta falta de constancia en la calidad del recurso ecuatoriano no se da solo por la falta de cultivos; sino también por la formación de un cuello de botella a la hora de seleccionar los granos, agudizado por los tiempos de clasificación a manos de agricultores y la poca innovación tecnológica en las industrias cacaoteras.

Las Redes Neuronales Convolucionales durante las últimas décadas han sido consideradas como una de las herramientas más poderosas para el procesamiento masivo de datos, al mismo tiempo se han popularizado ante las personas que se interesen por la inteligencia artificial, machine learning y deep learning (Aprendizaje automático y aprendizaje profundo).Ya que son capaces de manejar una gran cantidad de datos [3] agilizando y perfeccionando el sistema de clasificación.

El interés por tener capas ocultas más profundas ha comenzado, recientemente, a superar el rendimiento de los métodos clásicos en diversos campos. Especialmente en el reconocimiento y diferenciación de patrones que permiten una clasificación optima guiada por el aprendizaje automático [3]. Se propone una Red Neuronal Convolucional (CNN) que, toma su nombre de una operación matemática lineal entre matrices llamada convolución, debido a que posee varias capas, incluida la capa convolucional y la capa completamente conectada(Fully Conected) [4].

Este tipo de redes neuronales permite un desempeño destacado capaz de resolver problemas a través de la transferencia de aprendizaje, especialmente con aquellas operaciones que involucran el procesamiento de bases de datos de imágenes (ImageNet) y visión por computadora (Computer Vision) [3] para su posterior clasificación.
El siguiente trabajo de integración curricular aborda una investigación en torno a las estrategias y posibles alternativas que solucionen el déficit de calidad del grano de cacao en el Ecuador; debido a que el territorio ecuatoriano posee 37 000 hectáreas sembradas [5] lo que complica su proceso de clasificación en torno a tiempo y esfuerzo. Principalmente, se espera que la Red Neuronal Convolucional para la clasificación de granos de cacao con pulpa pueda analizar los patrones y las imágenes de cada una de los granos de cacao y los clasifique automáticamente con el mínimo margen de error y en un tiempo corto.

1.1. OBJETIVO GENERAL

Implementar un sistema de clasificación de pulpa de cacao fresco, basado en redes neuronales convolucionales bajo un esquema de transferencia de aprendizaje

1.2. OBJETIVOS ESPECÍFICOS

- Estudiar las redes neuronales convolucionales y el aprendizaje automático.
- Implementar una red neuronal convolucional como clasificador y extractor de características.
- Comparar, en términos de exactitud, la red neuronal convolucional con el proyecto de titulación basado en máquina de vectores de soporte(SVM)

1.3. ALCANCE

Para este clasificador primero se extraerá las imágenes RGB (Red Green Blue) de una base de datos provista por la universidad UTEQ en la facultad de ingeniería mecánica, el cual se encuentra disponible en la referencia [6].
Estas imágenes están clasificadas en tres clases, las cuales son:

- Excelente Calidad

- Media Calidad
- Mala Calidad

Esto se logró gracias a los conocimientos organolépticos que poseen los agricultores del cacao. En este trabajo de titulación se van a utilizar las 3 clases anteriormente citadas.

Figura 1.1: Cacao Excelente, Media y Mala Calidad

En la figura 1.2 Se muestra un diagrama de bloques de implementación del sistema de reconocimiento de calidad del cacao basado en redes neuronales convolucionales. Este modelo requiere de 2 fases: Entrenamiento y Prueba(Test).

En la fase de entrenamiento la base de datos pasará por un proceso de Data Augmentation para hacer nuestro sistema más robusto [7], luego esta nueva base de datos aumentada pasará por una red neuronal convolucional pre-entrenada, en la cual se utilizará transferencia de aprendizaje para ajustar los pesos de la misma.
En la fase de pruebas se escogerá una imagen que no ha sido utilizada en la fase de entrenamiento, está a su vez, ingresará al modelo validado, el cual la clasificara en una de las 3 categorías preestablecidas. Con esto calcularemos las métricas como la eficacia de entrenamiento, de validación y de pruebas.
Este proyecto de titulación no tiene producto final demostrable.

1.4. MARCO TEÓRICO

1.4.1. Redes Neuronales

De acuerdo a la Red Neuronal está conformada por el axion y éste se conecta con la dentrita por el proceso llamado sinapsis la misma que cambia en el proceso de aprendizaje las mismas que se pueden estructurar según el número de capas y según el grado de conexión. La técnica que dispone esta red es completar las tareas con indefinidas combinaciones la hacen única por su capacidad a datos incompletos, ambiguos o contradictorios, llegando así

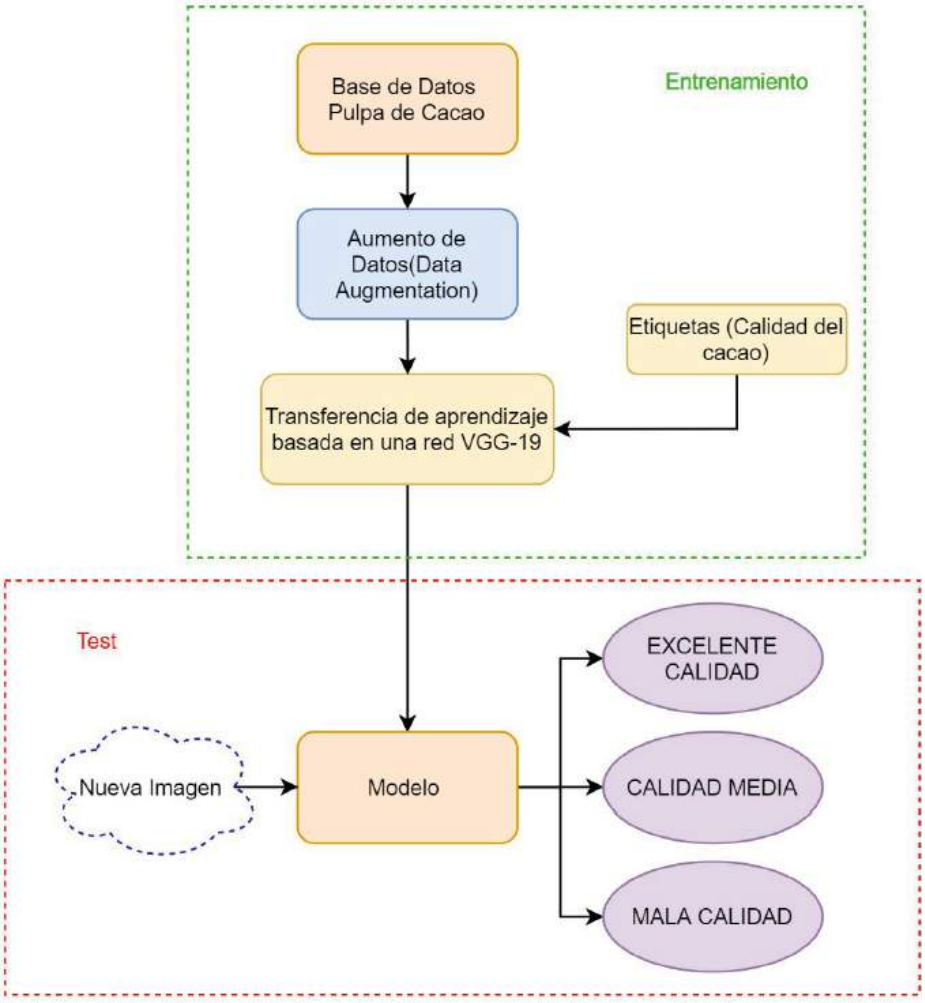


Figura 1.2: Diagrama de bloques de implementación del sistema de reconocimiento de calidad del cacao basado en redes neuronales convolucionales.

a un resultado deseado. Son sistemas computacionales inspirados en las neuronas biológicas lo cual constituye un conjunto de interconexiones que se conectan con las dendritas o entrada por medio de la sinapsis, la cual puede llegar cambiar durante el proceso de aprendizaje. Es una secuencia de algoritmos que investigan una conexión de conjunto de datos, ya que consta de nudos interconectados la cual tiene una similitud a una red neurológica biológica. Esta dispone de tres capas:

- Capa de entrada
- Capa oculta
- Capa de salida

Según el número de capas

Se dividen en:

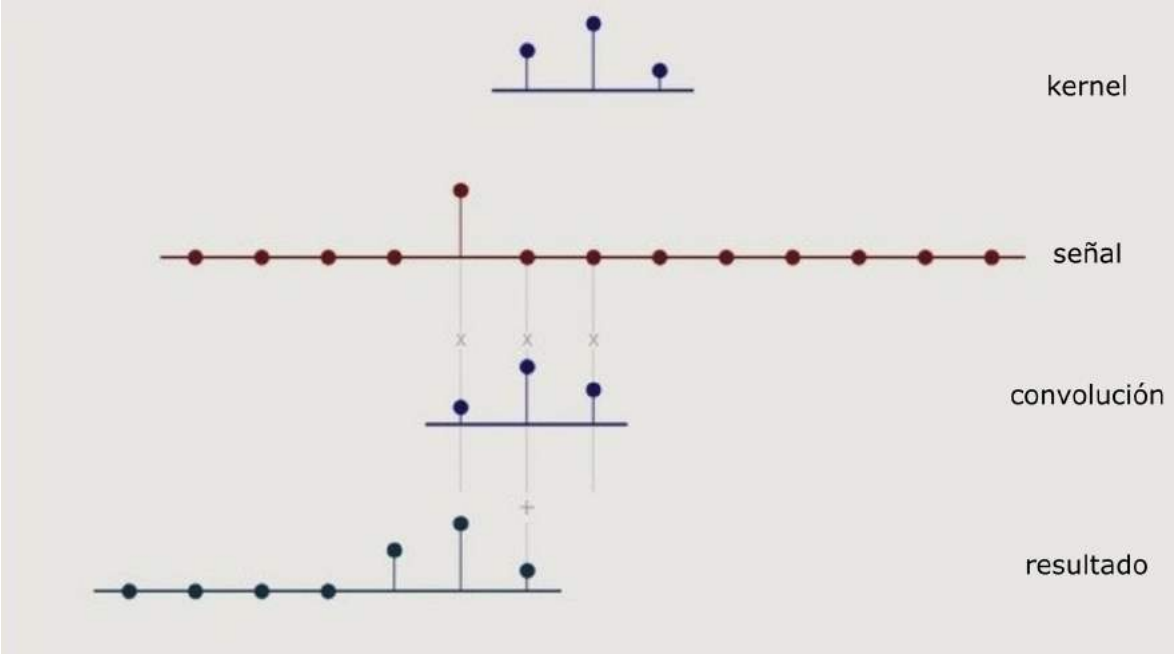


Figura 1.3: Red Neuronal Convolucional.

- Redes neuronales mono capas
- Redes neuronales multicapa

Redes neuronales monocapas:

Consta de una sola capa y su función es proyectar la capa de entrada a la capa de salida donde se realizan cálculos, esta red se utiliza generalmente para distorsionar señales.

Redes neuronales multicapa:

Consta de una serie de capas intermedias entre la capa de entrada y la capa de salida y pueden estar parcialmente o totalmente conectadas.

Según el tipo de conexiones

Se dividen en:

- Redes neuronales no recurrentes
- Redes neuronales recurrentes

Redes neuronales no recurrentes: En esta red se mueve solamente hacia adelante lo que ocasiona que no exista una retroalimentación de ningún tipo.

Redes neuronales recurrentes: Esta red permite retroalimentación utilizando lazos que pueden ser entre capas de distintas o de la misma neurona.

Según el grado de conexiones

Se dividen en:

- Redes neuronales totalmente conectadas
- Redes parcialmente conectadas
- Redes unidireccionales p de propagación hacia adelante (feedforward)
- Redes de programación hacia atrás (feedback)

Redes neuronales totalmente conectadas: Esta red se conecta con la capa anterior y la siguiente capa, generando redes concurrentes y no concurrentes. **Redes parcialmente conectadas:** En esta red se llega a una conexión parcial y nunca se llega a una conexión total ya que son de forma jerárquica.

Redes unidireccionales p de propagación hacia adelante (feedforward): En esta red ninguna salida neuronal se convierte en entrada, la información circula solo en un sentido.

Redes de programación hacia atrás (feedback): En esta red las neuronas de salida si se pueden convertir en neuronas de entrada o de un mismo nivel.

1.4.1.1. Redes Neuronales

Las Redes Neuronales Artificiales (RNA) están basadas en las redes neuronales del cerebro humano; están formadas por elementos similares a la neurona biológica para realizar funciones básicas y los compuestos químicos detectables se van beneficiando con el uso de los matices de sensores de un solo gas logrando utilizar un conjunto de sensores a costos más bajos. El objetivo de la configuración de una Red Neuronal Artificial es la generación de los diferentes niveles de saturación de gas y la implementación de sistemas identificadores, una vez que se haya identificado el clasificador se pueden introducir diferentes entradas y modelos.

Son un modelo de computación inspirado en un gran número de elementos simples, que se encuentran conectadas y transmiten señales entre sí, para tener la capacidad de retener y agrupar hechos los cuales pueden intervenir en tiempo real. Este sistema pretende simular un funcionamiento nervioso biológico.

Es un sistema que se encuentra formadas por unidades básicas llamadas neuronas las cuales tienen conexión entre sí, que tienen como relación de entrada y salida estas se basan en

un sistema nervioso humano y su objetivo fundamental de los algoritmos es poder diferenciar datos del mundo real (imágenes, textos, voz), y a su vez permite clasificarlos, procesarlos y etiquetarlos según corresponda.

Un conjunto de neuronas conectadas entre sí, que realizan un proceso en conjunto sin tener una tarea específica para cada una de ellas, tratando de emular el comportamiento del cerebro humano, los cuales ayudan a realizar tareas cognitivas como la memorización de patrones ayudando así a la clasificación u optimización de recursos.

Basándose en una idea simple: las redes neuronales tienen un enfoque de transición tanto en las matemáticas y estadística el cual se ha ido desarrollando con el pasar de los años y métodos de estudio, encontrándose en si algunos parámetros para combinarlos y encontrar un cierto resultado en un tiempo preciso.

Las redes neuronales ya capacitadas se pueden llegar a utilizar ya precisamente para aplicar la combinación de un formato en específico, es decir la clasificación o predicción que mejor se ajuste a un determinado caso. Y es fundamental una cantidad importante de recursos de un ordenador para llevar a cabo una buena red neuronal como buenos resultados.

Cada sistema de red neuronal está compuesto por ciertos elementos como:

- Conjunto de procesadores elementales.
- Patrón de conectividad.
- Dinámica de actividades.
- Entorno donde opera.

Mediante estos elementos las redes neuronales, empiezan a resolver problemas que se les pasa por alto a los ordenadores tomando en cuenta la amplia variedad de tareas que son difíciles para la denomina programación estructurada.

Las Redes Neuronales Artificiales están formadas a través de interconexiones de redes las mismas que son de forma jerárquica para que permitan la interacción con el mundo, también son conocidas como un sistema de elementos simples interconectados los mismos que generan información ante algún estimulo externo, las mismas que permiten resolver problemas individuales o grupales con la ayuda de distintos métodos para las tareas donde se pueden clasificar, diagnosticar e identificar datos.

Las neuronas de entrada captan particularmente señales que provienen de archivos de almacenamiento de patrones de aprendizaje, con diferencia que las neuronas de salida envían su señal fuera del sistema una vez concluido el proceso de la información, en cambio las neuronas ocultas activan y difunden información dentro del sistema sin tener contacto alguno con la información procesada del exterior.
Las conexiones de las redes neuronales son recurrentes ya que están permiten una retroalimentación, mediante el uso de lazos están pueden llegar a ser neuronas de diferentes capas o de una sola, en cambio las redes no recurrentes son aquellas que no dan paso a una retroalimentación de ningún tipo y estas disponen de una sola dirección hacia adelante.
Los modelos de las redes neuronales tienen como base los modelos matemáticos que da como definición una función y también son asociados con algoritmos de aprendizaje:
La función de red es aquella que se encuentra basada en las interacciones de la neurona biológicas.
La función de aprendizaje es establecer el conjunto de observaciones el cual permite y facilita resolver las actividades ya expuestas por el ordenador.
De acuerdo a existen múltiples ventajas y desventajas de las Redes Neurológicas Artificiales:

Ventajas

- El aprendizaje es adaptativo donde se puede aprender a partir datos los mismos que son representada como entradas.
- Existe cierta tolerancia a fallos parciales que pueden ocurrir pero no se pierde la información ya que este sistema funciona como el cuerpo humano.
- Para aprovechar la capacidad dela Red Neuronal Artificial se puede ejecutar a través de dispositivos de hardware especializados o por una computadora.

Desventajas

- Tienen dificultad para el aprendizaje de tareas grandes.
- De acuerdo al número de patrones a reconocer y la flexibilidad que exista para reconocer patrones es mayor el tiempo de aprendizaje.

- No tiene la capacidad para darle significado a los resultados.

La Red Neuronal Convolucional (RNC) cuenta con una estructura básica de tres capas: una capa de convolución, una de rectificación lineal y una llamada pooling, esta red puede incluir los núcleos que desee y en cualquier orden pero si es muy necesario establecer parámetros básicos para cada núcleo; las características de los parámetros se determinan una vez que se haya incluido un tamaño, ancho, alto y profundidad estándar de las entradas; es decir todas las imágenes deben medir lo mismo.
De acuerdo a los tipos de redes Neuronales Artificiales son:

Red backpropagation

Esta red aprende de la asociación de los patrones existentes, el backpropagation generaliza lo que es un comportamiento y lo realiza mediante patrones de entrenamiento y al ser esta como un cerebro humano necesita de una etapa de aprendizaje; es decir, requiere de una búsqueda de función de comportamientos acoplados al comportamiento del sistema teniendo en cuenta un valor mínimo de error.

Perceptor multicapa

Tiene algunos aspectos importantes como:

- Tiene células como conjunto de unidades de procesamiento.
- Existen conexiones entre las neuronas, pero cada conexión fija el efecto de cada señal.
- Cuenta con un estado de activación la misma que determina cual es la salida de la neurona.
- La regla propagación determina la entrada efectiva de una neurona a partir de las entradas existentes.
- En su entorno existen señales de entrada con el que debe operar la neurona.

1.4.2. HERRAMIENTAS UTILIZADAS

1.4.2.1. Matlab

MATLAB o *LABORATORIO DE MATRICES* es un lenguaje de programación diseñado y desarrollado para ingenieros y estudiantes que se dedican al diseño y el análisis. [8] El eje de MATLAB es su lenguaje de programación, que ofrece a los usuarios una experiencia fantástica con las funciones. Las funciones de MATLAB incluyen un lenguaje de cálculo basado

en matrices, lo que permite a los usuarios obtener la forma más natural de las matemáticas computacionales. El software es necesario para computar Big Data y calcular una ecuación fácil de entender. [8] Además, cuando hay una gran cantidad de datos, es responsabilidad del científico de datos analizarlos y llegar a algo único y fácil de entender. Por otra parte, las funciones son excelentes pero merecedoras para analizar datos críticos con la ayuda de un laboratorio de matrices.

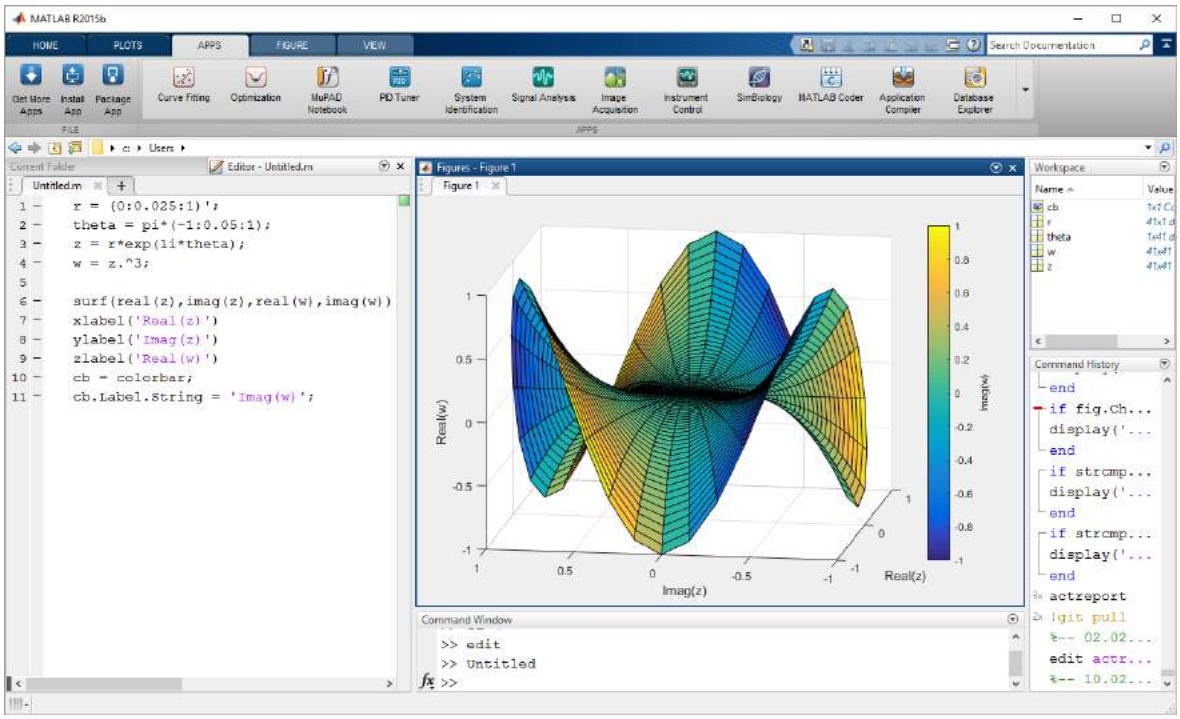


Figura 1.4: Matlab ejemplo de códigos e imágenes.

Las principales funciones de MATLAB son: [8]

- Analizar datos críticos
- Desarrollar algoritmos valiosos
- Crear modelos y aplicaciones de alto rendimiento
- Realizar álgebra lineal numérica
- Cálculo numérico de matrices
- Análisis y visualización de datos
- Trazado de gráficos para grandes conjuntos de datos

- Creación de interfaces para GUI y API

Las funciones de MATLAB son relativamente amplias, en comparación con otros programas informáticos. Es un lenguaje de alto nivel, óptimo para el cálculo numérico, la visualización y el desarrollo de aplicaciones básicas. Proporciona una amplia biblioteca de funciones matemáticas, como el álgebra lineal, la estadística, el análisis de Fourier, la optimización y el suministro de soluciones para las ecuaciones diferenciales ordinarias. Ofrece un entorno interactivo para la codificación como la exploración iterativa, el diseño y la resolución de problemas. [8] Tiene gráficos incorporados como el gráfico1.4 para visualizar datos y crear gráficos personalizados de MATLAB. Las interfaces de programación de MATLAB ofrecen un conjunto de herramientas para mejorar la calidad del código, la capacidad de mantenimiento y la medida del rendimiento. Además, las funciones integran los algoritmos con aplicaciones y lenguajes externos como C, Java, .Net y Microsoft Excel.

1.4.2.2. Python

Figura 1.5: Python logo.

Python se ha convertido en uno de los lenguajes de programación más populares del mundo en los últimos años. Es utilizado en todo, desde el aprendizaje automático hasta la construcción de sitios web y las pruebas de software. Lo pueden utilizar tanto desarrolladores como no desarrolladores.

Según Shein [9]: Python es uno de los lenguajes de programación más extendidos del mundo, en este lenguaje de programación se han creado diferentes software, desde el algoritmo de recomendación de Netflix hasta el software que controla los autos autodirigidos como los Tesla [9] .

Python es un lenguaje de propósito general, lo que significa que está diseñado para ser

utilizado en una serie de aplicaciones [10], estos son:

- Análisis de datos y aprendizaje automático
- Desarrollo web
- Automatización o scripting
- Pruebas y prototipos de software
- Lo que tu mente desee.

1.4.2.3. Google Colab

El google colab que también es conocido como *Colaboratory*, es un entorno que nos permite ejecutar y programar Python desde nuestro navegador.
Algunas de las ventajas que nos ofrece Google Colab son:

- Es muy fácil de realizar
- No necesita de ninguna reconfiguración
- Tenemos acceso gratuito a GPUs y TPUs
- Nos permite compartir nuestro contenido con otras personas en el mismo entorno Colab o fuera, de forma fácil.

Google Colab nos proporciona de una única GPU, esta es del modelo NVIDIA Tesla K80 de 12 GB tal como nos lo indica [11], que puede utilizarse hasta 12 horas de forma continua. Recientemente, Colab también ha empezado a ofrecer TPU gratis.

¿Qué es una GPU y una TPU?

GPU es una unidad de procesamiento gráfico, esta fue diseñada y utilizada originalmente para los gráficos 3D con el fin de acelerar cosas como el renderizado de vídeo [12], pero con el tiempo, su capacidad de cálculo paralelo las convirtió en una opción extremadamente popular para su uso en la Inteligencia Artificial.
En su lugar, las TPU son unidades de procesamiento tensorial, una arquitectura designada para aplicaciones de aprendizaje profundo *Deep Learning* o aprendizaje automático*Machine Learning*. [12]
Estas fueron inventadas por Google [12], como circuitos integrados de aplicación específica (ASIC), diseñados específicamente para manejar las demandas computacionales del aprendizaje automático y acelerar los cálculos y algoritmos de IA. Cuando Google diseñó la

TPU, creó una arquitectura de dominio específico. Lo que esto significa es que en lugar de diseñar un procesador de propósito general como una GPU o una CPU, Google lo diseñó como un procesador matricial especializado en cargas de trabajo de redes neuronales. Al diseñar la TPU como un procesador matricial en lugar de un procesador de propósito general, Google resolvió el problema de acceso a la memoria que ralentiza las GPU y las CPU y les obliga a utilizar más potencia de procesamiento.

2. METODOLOGÍA

En este apartado se describirá aquellos métodos y procesos que se realizaron al momento de clasificar, de manera específica, que la pulpa de cacao; tomando en consideración tres categorías: Excelente Calidad, Media Calidad y Mala Calidad.

Se utilizó el repositorio de imágenes de la Universidad de Quevedo, base de datos que permitió conocer los tipos de pulpas de cacao y su respectiva clasificación; realizada por los agricultores y sus conocimientos sobre sus propiedades organolépticas, permitiéndonos encaminar este proyecto de manera correcta y con las bases necesarias para lograr los resultados que nos hemos propuesto.

La metodología utilizada alrededor de todo el trabajo se profundiza en la figura 1.2.

2.1. BASE DE DATOS

Para entrenar nuestra Red Neuronal Convolucional utilizamos la base de datos de cacao de la Universidad Técnica de Quevedo [6], la cual consta de un repositorio completo de 259 imágenes. Cada una de estas imágenes será tratada por separado en diferentes etapas para lograr sacar el mayor provecho posible. Estas imágenes han sido clasificadas gracias a sus propiedades organolépticas por agricultores expertos, los cuales las han clasificados en 3 grandes grupos.

- Excelente Calidad
- Media Calidad
- Mala Calidad

2.1.1. CALIDAD

Para que los productores de cacao puedan vender su producto conocido como cacao en ”baba”. Deben pasar un estricto control del grano y su mucílago. Este mucílago, también conocido como la piel del cacao, es una pulpa viscosa, con textura de algodón, el cual recubre el grano de cacao. [13] Para obtener un grano fresco de calidad los agricultores han desarrollado ciertas habilidades, que les permiten clasificar cada pulpa de cacao. Como lo

dice [14], entre los parámetros que se toman en cuenta para la calidad del grano de cacao están:

- La pulpa de cacao debe ser lo más fresca, de ser posible extraída el mismo día.
- Cada mazorca de cacao debe ser madura, impecable, sana y no debe tener ningún tipo de daño.
- Ninguno de los granos de la pulpa deben ser de color negro, café o traslucidos; no deben permitir pasar el color del grano.

Como referencia a lo citado anteriormente, presentaremos una muestra de cada una de las calidades del cacao, tomando en cuenta el estado del grano.
En la figura 2.1 podemos encontrar la mejor calidad de cacao; en la figura 2.3 la peor calidad de cacao, y la figura 2.2 es una calidad promedio.

Figura 2.1: Pulpa de cacao de Buena Calidad

2.2. ARQUITECTURA DE LA CNN

La arquitectura de la red neuronal convolucional que nosotros desarrollamos consta de varias capas o layers, cada una de estas capas tiene una función especifica. En el gráfico de la Figura 2.4 podemos observar como ingresa cada una de las imágenes de la base de datos y es procesada referencialmente.
Nuestra red neuronal convolucional ha sido desarrollada gracias a la transferencia de aprendizaje ”TRANSFER LEARNING”, por lo que obtenemos un modelo base basados en una red VGG19, esta red VGG19 utiliza los pesos del modelo imagenet.

Figura 2.2: Pulpa de cacao de Media Calidad

Figura 2.3: Pulpa de cacao de Mala Calidad

Este modelo *imagenet* que cuenta con mas de 14 millones de imágenes en su base de datos, 21000 grupos en los que están clasificados (synsets) y un poco más de 1 millón de imágenes que tienen anotaciones que poseen cuadro delimitador, estos son cuadros que están alrededor de objetos identificados en las imágenes). [15]

La red Neuronal VGG19 es una red neuronal realizada y planificada por K. Simonyan y A. Zisserman, esta red neuronal ya ha sido pre-entrenada con el modelo imagenet, esto nos ahorra mucho tiempo y muchos recursos.
Para sacar ventaja del *Transfer Learning*, sacamos un modelo base basado en la red neuronal VGG19, recuperamos el peso de cada una de las capas convolucionales, añadimos 2 capas mas, y a la final añadimos la última capa de clasificación. Todo lo dicho anteriormente lo realizamos en el segmento de código 2.1

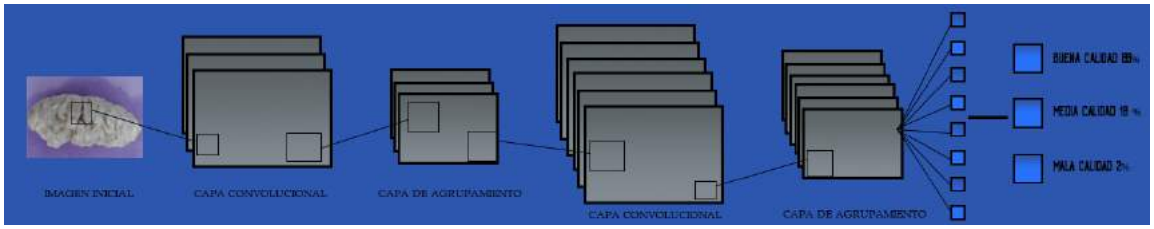


Figura 2.4: Red Neuronal Convolucional Referencial

Segmento de código 2.1: Arquitectura

```
"""Importo  VGG19 sin la ultima capa y añado capas densas las capas densas ...
    están activadas
con RELU y la ultima capa contiene el mismo numero de neuronas que de clases ...
    softmax"""
base_model=VGG19(weights='imagenet',include_top=False) #importamos el modelo ...
    con los pesos imagenet y descartamos la ultima capa.

x=base_model.output

x=GlobalAveragePooling2D()(x)
x=Dense(1024,activation='relu')(x)
#añadimos capas densas para que el modelo aprenda funciones mas complejas y ...
    clasifique con mejores resultados
x = Dropout(0.25)(x) # Realizamos un Dropout de 0.25
x=Dense(512,activation='relu')(x) #esta capa igual tiene activación relu
preds=Dense(2,activation='softmax')(x) #Capa final con activación softmax

#Indicamos las entradas y salidas de nuestro modelo
model=Model(inputs=base_model.input,outputs=preds)

# Especificamos entradas
# Especificamos las salidas
# Así ya queda creado nuestro modelo
```

2.3. PREPROCESAMIENTO

Para entrenar nuestra *RED NEURONAL CONVOLUCIONAL* utilizamos la base de datos provista por la Universidad Técnica Estatal de Quevedo (UTEQ), esta base de datos, cuenta con 259 imágenes separadas en las tres clases citadas anteriormente.

2.3.1. DATA AUGMENTATION

Cuando entrenamos una red neuronal lo que nosotros realmente realizamos es ajustar para que esta red neuronal pueda clasificar y etiquetar en este caso una imagen de entrada, y de salida la clase a la que pertenece esta. Para optimizar esto nosotros realizamos *DATA AUGMENTATION* o aumento de datos por su traducción al español, esto nos permite que con ciertos cambios podamos tener una base datos mas extensa que la anterior. El método que nosotros realizamos fue el de *Tiling*, el cual consiste en agarrar la imagen inicial y separar en cuadros ”tiles”de la misma dimensión lo que nos da muchos mas datos para que nuestra red neuronal se nutra y aprenda.

Figura 2.5: Imagen Original Base de Datos.

Figura 2.6: Imagen sin fondo.

Para que esto se realice primero creamos un segmento de código para quitar el fondo a cada una de las imágenes de la base de datos. Por lo que, la imagen 2.5 que es la original de la base de datos , es tratada hasta llegar al resultado de la imagen 2.6.

Como podemos revisar en la figura 2.6 en este caso partimos una imagen previamente sin fondo, a esta imagen la partimos en tiles y así terminamos con el proceso de *”DATA AUGMENTATION”*. Todo el proceso mas gráficamente lo podemos observar en la figura 2.7.

Para realizar este proceso denominado ”Tiling”se implemento el segmento de código 2.2.

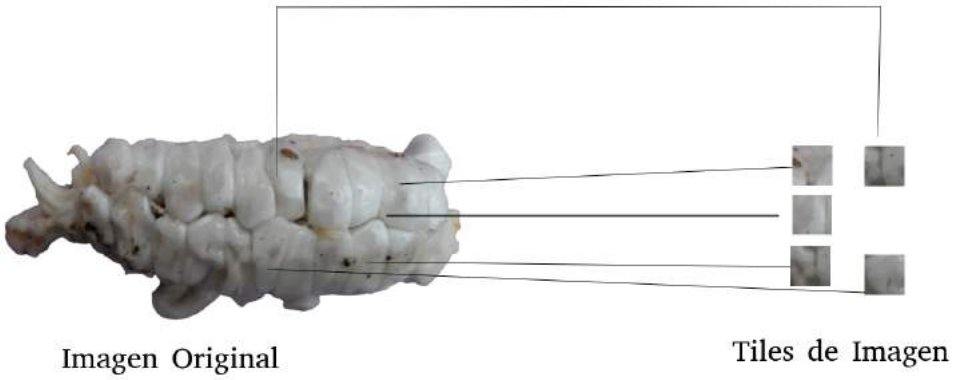


Figura 2.7: Proceso de Data Augmentation.

Segmento de código 2.2: Image Tiling

```
import cv2
import pandas as pd
path_to_img = ...
    "/home/paul/Descargas/FotosCacaoRecortado/Fotos/Mediacalidad/7/SAM_1553.jpg"
img = cv2.imread(path_to_img)
img_h, img_w, _ = img.shape
split_width = 200
split_height = 200

df = pd.DataFrame()



def start_points(size, split_size, overlap=0):
    points = [0]
    stride = int(split_size * (1-overlap))
    counter = 1
    while True:
        pt = stride * counter
        if pt + split_size >= size:
            points.append(size - split_size)
            break
        else:
            points.append(pt)
        counter += 1
    return points


```

```
X_points = start_points(img_w, split_width, 0)
Y_points = start_points(img_h, split_height, 0)

count = 1
name = 'SAM_1551_splitted'
frmt = 'jpeg'

for i in Y_points:
    for j in X_points:
        split = img[i:i+split_height, j:j+split_width]
        cv2.imwrite('{}_{}.{}'.format(name, count, frmt), split)
        df = df.append({'Posicion X': i, 'Posicion Y':j}, ignore_index=True)
        count += 1


l= df.to_csv("Coords71.csv")
```

Como podemos observar en el segmento de código 2.2, luego de realizar el proceso de tiling, guardamos las coordenadas de cada una de las imágenes en un archivo de formato *CSV*, que luego puede leerse para el momento de la superposición de resultados; luego lo explicaremos detalladamente en los capítulos siguientes.

Para revisar que todas las capas están correctamente estructuradas, y si van a ser entrenadas o no; utilizamos el segmento de código 2.3. Este código nos permite imprimir cada una de las capas y si se las va a entrenar o no.

Segmento de código 2.3: Número de capas

```

"Para saber  como va la arquitectura de nuestro modelo utilizamos el ...
    siguiente comando"


for layer in model.layers[:1]:
    layer.trainable=False
for layer in model.layers[1:]:
    layer.trainable=True

for i,layer in enumerate(model.layers):
    print(i,layer.name,layer.trainable)

```

```
13 # Esto  nos permite imprimir el numero de capas, y saber exactamente cuales ...
       se estan entrenando y cuales no.
```

2.4. ENTRENAMIENTO Y VALIDACIÓN

Antes de realizar el entrenamiento de la *Red Neuronal Convolucional*, tenemos separadas cada una de las imágenes ya en su set de datos aumentados. Cada una de estas entradas que van a ser entrenadas fueron separadas aleatoriamente en sus respectivas carpetas y en los porcentajes indicados a continuación.

- Entrenamiento: 80 %
- Validación: 10 %
- Test o Prueba: 10 %

Para entrenar la red neuronal utilizamos el segmento de código. 2.4

Segmento de código 2.4: Arquitectura de Entrenamiento CNN

```
1
2  # Cargamos desde el directorio nuestra información de entrenamiento hacia un ...
       data generator.
3
4  train_datagen=ImageDataGenerator(preprocessing_function=preprocess_input)
5
6  train_generator=train_datagen.flow_from_directory('/home/paul/Descargas/FotosCacaoRecortado/
7                                                    target_size=(224,224),
8                                                    color_mode='rgb',
9                                                    batch_size=64,
10                                                   class_mode='categorical',
11                                                   shuffle=True)
12
13 # Cargamos desde el directorio nuestra información de validación hacia un ...
       data generator.
14 valid_datagen=ImageDataGenerator(preprocessing_function=preprocess_input)
15 valid_generator=valid_datagen.flow_from_directory('/home/paul/Descargas/FotosCacaoRecortado/
16                                                   target_size=(224,224),
17                                                   color_mode='rgb',
18                                                   batch_size=32,
```

```
                                                    class_mode='categorical',
                                                    shuffle=True)

# Cargamos desde el directorio nuestra información de prueba  hacia un data ...
    generator.
test_datagen=ImageDataGenerator(preprocessing_function=preprocess_input)
test_generator=test_datagen.flow_from_directory('/home/paul/Descargas/FotosCacaoRecortado/Fo
                                                    target_size=(224,224),
                                                    color_mode='rgb',
                                                    batch_size=4,
                                                    class_mode='categorical',
                                                    shuffle=False)

#Compilamos nuestro modelo con un optimizador tipo "Adam".
#Las perdidas las hacemos por categorical cross entropy.
#Para evaluar nosotros utilizaremos la exactitud como métrica.

model.compile(Adam(lr=0.0001),loss='categorical_crossentropy',metrics=['accuracy'])
# loss function will be categorical cross entropy
# evaluation metric will be accuracy
# Declare the filepath for the saved model
filepath = "modelo3clasesPruebaRecortadaaumentadasplit2_VGG19.h5"


# Declaramos un checkpoint el cual nos permite ir guardando el modelo
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1,
                             save_best_only=True, mode='max')
# Esto lo utilizamos para reducir el LR
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5, patience=4,
                              verbose=1, mode='max', min_lr=0.00001)
callbacks_list = [checkpoint, reduce_lr]
callbacks_list_early =[checkpoint, EarlyStopping(monitor='val_loss', ...
    min_delta=0, patience=10)]

# calculamos los pasos al dividir el numero de datos de entrada vs el batch ...
    size

step_size_train=train_generator.n//train_generator.batch_size

step_size_val=valid_generator.n//valid_generator.batch_size

step_size_test=test_generator.n//test_generator.batch_size
```

```
58
59 Realizamos el acople del generador con el siguiente código.
60
61 hist= model.fit_generator(generator=train_generator,
62                     steps_per_epoch=step_size_train,
63                     validation_data=valid_generator,
64                    validation_steps=step_size_val,
65                     epochs=100,
66                     callbacks=callbacks_list)
```

Tal como podemos ver en el segmento de código 2.4, lo que primero realizamos nosotros es ingresar los datos que tenemos de entrada en nuestro generador de datos ”DATA GENERATOR”. Cada uno de estos datos ingresan en orden desde el directorio preestablecido. En nuestro caso el directorio es el de descargas */Descargas/FotosCacaoRecortado/Fotos*.
Al compilar nuestro modelo utilizamos el optimizador **Adam**. La optimización de Adam es un método de descenso de gradiente estocástico que se basa en la estimación adaptativa de momentos de primer y segundo orden.
Según Kingma [16], el método es *computacionalmente eficiente, requiere poca memoria, es invariable al cambio de escala diagonal de gradientes y es adecuado para problemas que son grandes en términos de datos y parámetros*.
Este tal y como nos indica su librería [17] tiene argumentos:
La tasa de aprendizaje *learning rate*: Un valor fijo, un valor de punto flotante o un programa que esta basado en la librería *tf.keras.optimizers.schedules.LearningRateSchedule*, o un numero que puede ser invocado que no toma argumentos y devuelve el valor real para usar. El valor predeterminado es 0.001, en nuestro caso utilizamos la décima parte: 0.0001.
Pérdidas: Utilizamos esta función de pérdida de entropía cruzada *Categorical cross entropy* ya que tenemos dos o más etiquetas, este tipo de p’erdidas nos ayudan a calcular la pérdida de entropía cruzada entre las etiquetas y sus predicciones [18].
Métricas: La cual nos proporciona una lista de métricas que serán evaluadas por el modelo durante el entrenamiento y la prueba [19] .Cada una de ellas puede ser una secuencia (nombre de una función incorporada), una función o una instancia de tf.keras.metrics [20].
Utilizamos en nuestro segmento de código 2.4 una lìnea de codigo llamada Checkpoint , este Checkpoint nos permite guardar el mejor modelo.

Segmento de código 2.5: Checkpoint para guardar el mejor modelo

```
checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1,
                             save_best_only=True, mode='max')
```

Que quiere decir esto?

Que esta variable irá analizando nuestros resultados de precisión en validación, y guardará el modelo que mejor precisión obtenga.

Este modelo se lo guardara como:

También usamos una pequeña linea de código a la cual la denominamos **reduce_lr**, esta linea de código va monitorizando todos nuestros resultados de precisión en validación y lo va reduciendo, si es o no necesario para obtener mejores niveles de precisión y un resultado mas acertado [20].

Segmento de código 2.6: Reducir la tasa de aprendizaje

```
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5, patience=4,
                              verbose=1, mode='max', min_lr=0.00001)
```

Cálculo de los pasos: Para calcular los pasos que va a dar nuestro modelo generador usamos nuestro Step Size o tamaño de pasos, el cual calculamos al dividir el tamaño total de nuestras muestras, ya sean estas de entrenamiento, validación o prueba; con nuestro batch size o tamaño de lote previamente seleccionado. [19]

El tamaño del lote determina el número de muestras en cada mini lote. Su máximo es el número de todas las muestras, lo que hace que el descenso por gradiente sea preciso, la pérdida disminuirá hacia el mínimo si la tasa de aprendizaje es lo suficientemente pequeña, pero las iteraciones son más lentas.

Su mínimo es 1, lo que hace que el descenso de gradiente sea estocástico: Rápido pero la dirección del paso de gradiente se basa sólo en un ejemplo, la pérdida puede saltar. El tamaño del lote permite ajustar entre los dos extremos: dirección precisa del gradiente e iteración rápida. Además, el valor máximo de batch size puede ser limitado si nuestro modelo o conjunto de datos no cabe o da problemas en la memoria disponible de la GPU.

Keras fit_generator:

Ya que los datos de nuestro modelo es amplio y pesado (imágenes), optamos por la opción fit_generator ya que utilizamos iteradores los cuales nos permiten ir cargando dato por

dato y así lograr que nuestra memoria, ya sea usando GPU o TPU no se nos bloquee. El fit_generator tiene varias entradas, en nuestro caso utilizamos estas:

Tipo de generador: En nuestro caso utilizamos nuestro generador de entrenamiento en el cual previamente se cargo la carpeta en la que se encuentran todas las imágenes de entrenamiento.

Pasos por época: Estos fueron los que se calcularon anteriormente con la ayuda de nuestro batch size.

Datos para validación: Aqui utilizamos nuestro generador de validación en el cual previamente se cargo la carpeta en la que se encuentran todas las imágenes de validación.

Pasos para validación: Estos fueron los que se calcularon anteriormente con la ayuda de nuestro batch size de validación.

Número de épocas: Es el número de épocas en los que queremos entrenar nuestro modelo.

Callbacks: Es nuestra lista de funciones que van a ser llamadas en cada época durante el entrenamiento de nuestro modelo, en nuestro caso fueron el **Checkpoint** 2.5 y el **reduce_lr** 2.6.

Segmento de código 2.7: Evaluación del modelo

```
# Evaluación del modelo
# Evaluación de la ultima epoca
val_loss, val_acc = \
model.evaluate_generator(test_generator, steps=step_size_test)
""
""
""
""
""
""
""
print('val_loss:', val_loss)
print('val_acc:', val_acc)


#Aqui empieza la carga del modelo y validacion
# Evaluacion de la mejor epoca
```

```
18 model.load_weights('modelo3clasesPruebaRecortadaaumentadasplit1_VGG19.h5')
19
20 val_loss, val_acc, = \
21 model.evaluate_generator(valid_generator, steps=step_size_val)
22
23 print('val_loss:', val_loss)
24 print('val_acc:', val_acc)
```

En nuestro nuevo segmento de código 2.7 empezamos la validación de nuestro modelo. Las primeras líneas de código nos permiten primero asignar a nuestras etiquetas de pérdida y exactitud los nuevos valores que se obtienen, a estos los hemos llamado *val_loss y val_acc* respectivamente.
Para esta fase cargamos los pesos del "Mejor modelo', para esto usamos la función **model.load_weights**, este nos permite cargar los pesos de cada una de las capas desde nuestro archivo guardado como: ***modelo3clasesPruebaRecortadaaumentadasplit2_VGG19.h5***

Volvemos a asignar el mismo nombre a nuestros datos de pérdida y exactitud. En la linea 21 de nuestro segmento de código 2.7 realizamos la evaluación de nuestro modelo, para esto utilizamos ***model.evaluate_generator***, en nuestro caso únicamente haremos uso de dos entradas:

Dato de entrada : En este caso sería nuestro generador que nos va ingresando cada uno de los datos y las clases a las que pertenece.

Pasos: Es el número total de pasos que va a dar nuestro modelo para evaluar, nosotros previamente calculamos los pasos y esos los ponemos.

A estos valores los enviamos a imprimir en pantalla para ir observando mientras se esta ejecutando nuestro código, tal como lo observamos en el gráfico 2.8.

```
Epoch 1/100
45/49 [==========================>...] - ETA: 2:06 - loss: 0.2064 - accuracy: 0.9038
```

Figura 2.8: Ejemplo de exactitud y pérdidas.

2.5. EVALUACIÓN

Existen muchas formas para evaluar el desempeño de nuestra Red Neuronal Convolucional [21], nosotros realizamos las mas importantes o especificas para nuestro caso.

Matriz de Confusión

En nuestro caso primero vamos a realizar una matriz de confusión esta matriz es un resumen de los resultados de las predicciones en un problema de clasificación. El número de predicciones correctas e incorrectas se desglosa por cada clase de nuestro modelo . Esta es la clave de la matriz de confusión. Esto nos permite saber donde cual es la clase en la que tenemos mas errores. Así podemos tener una idea mas clara de como esta el desempeño de la red.

Segmento de código 2.8: Matriz de confusion

```
1  def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion ...
       matrix', cmap=plt.cm.Blues):
2      """
3      This function prints and plots the confusion matrix.
4      Normalization can be applied by setting `normalize=True`.
5      """
6      if normalize:
7          cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
8          print("Normalized confusion matrix")
9      else:
10         print('Confusion matrix, without normalization')
11
12     print(cm)
13     plt.figure(1)
14     plt.imshow(cm, interpolation='nearest', cmap=cmap)
15     plt.title(title)
16     plt.colorbar()
17     tick_marks = np.arange(len(classes))
18     plt.xticks(tick_marks, classes, rotation=45)
19     plt.yticks(tick_marks, classes)
20
21     fmt = '.2f' if normalize else 'd'
22     thresh = cm.max() / 2.
23     for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
24         plt.text(j, i, format(cm[i, j], fmt),
```

```
                    horizontalalignment="center",
                    color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()

cm = confusion_matrix(test_labels, predictions.argmax(axis=1))

cm_plot_labels = ['Excelente', 'Mala','Media']

plot_confusion_matrix(cm, cm_plot_labels)
```

El segmento de código 2.8 nos permite mostrar las formas en que nuestro modelo de clasificación se confunde cuando hace predicciones.

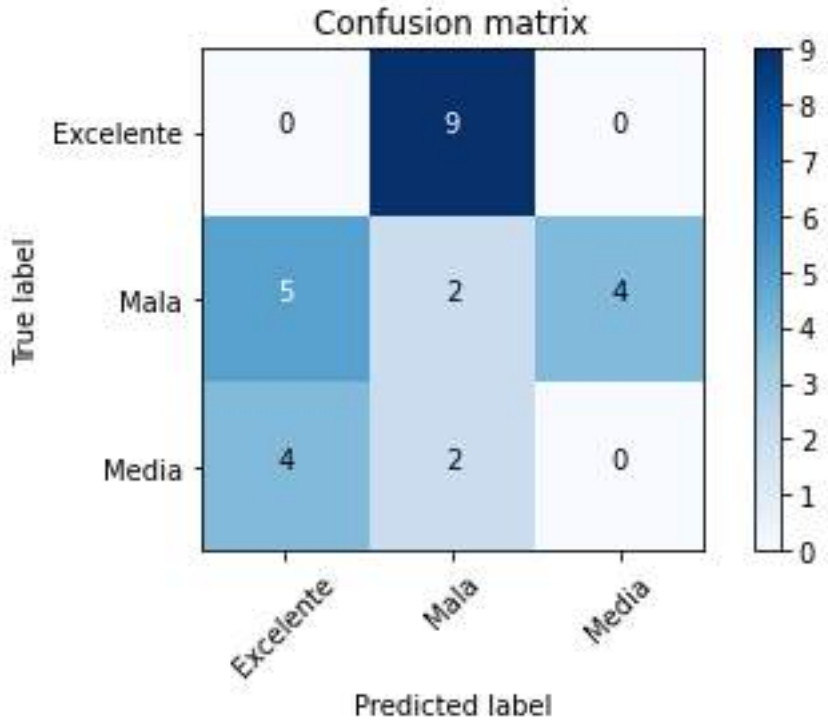


Figura 2.9: Ejemplo de un resultado de nuestra matriz de confusión

Curvas de precisión Las curvas de aprendizaje son una de las herramientas más utilizadas para hacer un diagnóstico de los problemas, como el underfitting o el overfitting de los modelos, así como para la comprobar o depurar el código y las implementaciones.

Segmento de código 2.9: Curvas de Aprendizaje

```
plt.figure(2)
fig, ax1= plt.subplots()
color = 'tab:red'
ax1.set_ylabel('Exactitud', color='tab:red')
ax1.plot(hist.history["val_accuracy"], color='tab:red')
ax1.tick_params(axis='y',labelcolor='tab:red')
ax2= ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Perdidas', color='tab:blue')
ax2.plot(hist.history["val_loss"], color='tab:blue')
ax1.tick_params(axis='y',labelcolor='tab:blue')
fig.tight_layout()

plt.figure(3)
fig, ax1= plt.subplots()
color = 'tab:red'
ax1.set_ylabel('Exactitud Val', color='tab:red')
ax1.plot(hist.history["val_accuracy"], color='tab:red')
ax1.tick_params(axis='y',labelcolor='tab:red')
ax2= ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Exactitud Test', color='tab:blue')
ax2.plot(hist.history["test_accuracy"], color='tab:blue')
ax1.tick_params(axis='y',labelcolor='tab:blue')
fig.tight_layout()

plt.figure(4)
fig, ax1= plt.subplots()
color = 'tab:red'
ax1.set_ylabel('Exactitud Test', color='tab:red')
ax1.plot(hist.history["test_accuracy"], color='tab:red')
ax1.tick_params(axis='y',labelcolor='tab:red')
ax2= ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Exactitud Train', color='tab:blue')
ax2.plot(hist.history["train_accuracy"], color='tab:blue')
ax1.tick_params(axis='y',labelcolor='tab:blue')
fig.tight_layout()

plt.figure(5)
fig, ax1= plt.subplots()
```

```
color = 'tab:red'
ax1.set_ylabel('Exactitud train', color='tab:red')
ax1.plot(hist.history["train_accuracy"], color='tab:red')
ax1.tick_params(axis='y',labelcolor='tab:red')
ax2= ax1.twinx()
color = 'tab:blue'
ax2.set_ylabel('Perdidas train', color='tab:blue')
ax2.plot(hist.history["train_loss"], color='tab:blue')
ax1.tick_params(axis='y',labelcolor='tab:blue')
fig.tight_layout()
plt.show()
```

En el segmento de código 2.9 se realizo la implementación de 4 gráficos, en estos podemos ver las siguientes curvas de aprendizaje:

- Exactitud en validación vs Pérdidas en validación
- Exactitud en validación vs Exactitud en Prueba
- Exactitud en Prueba vs Exactitud en Entrenamiento
- Exactitud en Prueba vs Pérdidas en Prueba

Un ejemplo de estos gráficos se puede observar en el gráfico 2.10, a medida que exponemos el modelo de forma incremental durante el proceso de entrenamiento a ejemplos del conjunto de entrenamiento y actualizamos los parámetros del modelo, calculamos una pérdida para cada lote incremental de ejemplos como parte del proceso de descenso de gradiente. Esta pérdida, sólo se calcula de un lote , es una estimación de la pérdida en todo el conjunto de entrenamiento. Esta es libre y no requiere tiempo adicional para calcularla. Calcular la pérdida en un conjunto de entrenamiento grande suele llevar mucho tiempo. Así que no podemos permitirnos hacerlo con mucha frecuencia. Dado que no obtenemos la pérdida de validación de forma libre durante el entrenamiento, tenemos que calcularla periódicamente. Por ejemplo, después de cada barrido a través de los datos de entrenamiento, lo que se denomina una época. Cuando la pérdida de validación empieza a subir, es cuando ocurre el sobre ajuste conocido como overfitting.

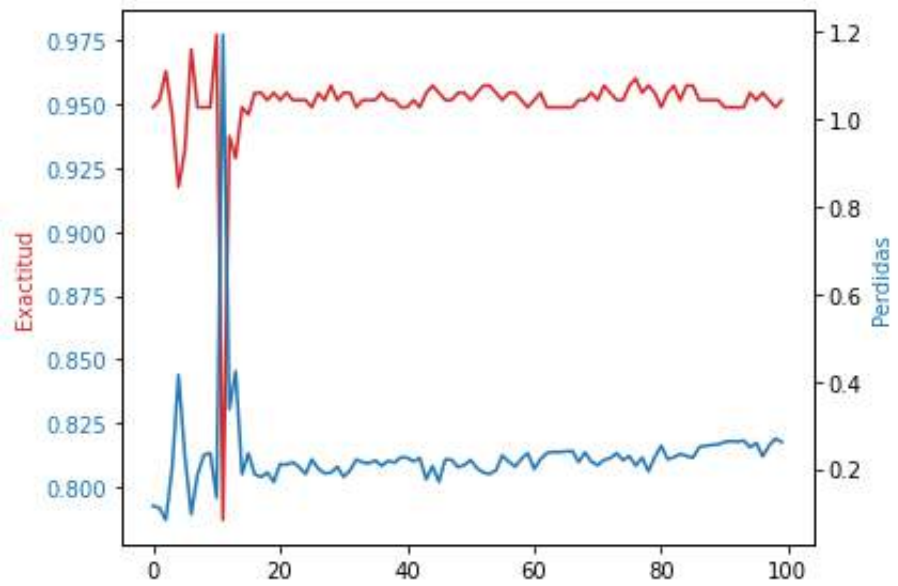


Figura 2.10: Ejemplo de una Curva de Aprendizaje Exactitud vs Pérdidas Validación.

K Fold Cross Validation

Ya que dividimos los datos en conjuntos de entrenamiento, validación y prueba. El K fold Cross Validation consiste en lugar de crear sólo una división, dividimos los conjuntos de datos en K subconjuntos (pliegues) y luego entrenamos K-1 de ellos. El último de los subconjuntos nosotros lo utilizamos como conjunto de prueba. Al hacer esto una y otra vez, podemos obtener muchas estimaciones del modelo en una variedad de submuestras de los datos esto lo podemos ver mas gráficamente. La validación cruzada es un método de remuestreo que puede tener una ventaja cuando no tenemos muchos datos como en nuestro caso.

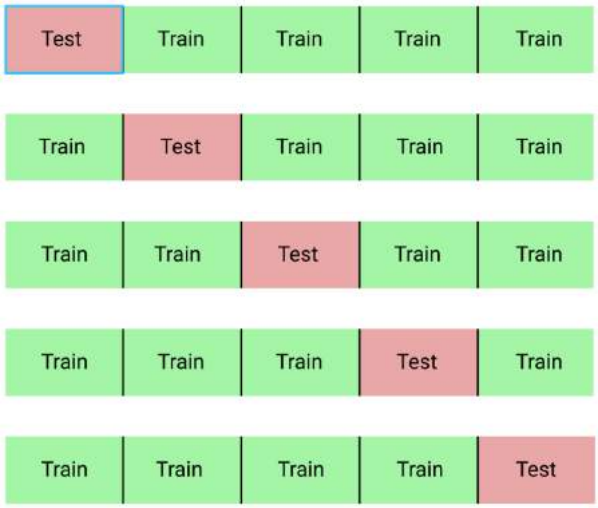


Figura 2.11: Ejemplo de como realizar K Fold Cross validation

Método T-SNE

El método ”t-distributed Stochastic Neighbor Embedding”(t-SNE) es un método de reducción de la dimensionalidad, utilizado principalmente para la visualización de datos en mapas 2D y 3D. Este método puede encontrar conexiones no lineales en los datos y por eso es muy popular. Cuando se trabaja con datos con más de 2 ó 3 características, es posible que se quiera comprobar si los datos tienen clusters. Esta información nos da una visualización para poder entender sus datos y, si es necesario, tomar ciertos tipos de de modelos como el k-means.

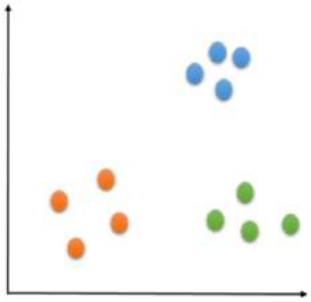


Figura 2.12: Ejemplo de la utilización de TSNE

Para obtener los mejores resultados evaluamos nuestro clasificador como uno de dos clases, este clasificador nos da resultados que son exactos y precisos. Luego de que nuestro clasificador previamente entrenado ya ha sido cargado, procedemos a analizar imágenes nuevas, estas son las que nos indicaran los resultados de todas las imágenes.

Segmento de código 2.10: Código implementado para cada Imagen

```
1  # Realizamos las predicciones
2  predictions = model.predict(test_generator, steps=step_size_test, verbose=1)
3
4
5  suma=0
6
7  i=-0
8  a=len(predictions)
9
10 # b= pred[6,0]
11
12 while i<a :
```

```
13      b= predictions[i,0]
14      suma= suma+b
15      i=i+1
16      #print('El segmento',i,'es ',b*100,'% de Buena Calidad y ', (1-b)*100,'% ...
            de Mala Calidad')
17
18  promedio=suma/a*100
19  mala=100-promedio
20
21  print('La imagen es',promedio,'%de Buena Calidad y un',mala,'%de Mala ...
        Calidad')
```

Esto lo podemos observar mejor en el segmento de código 2.10, primero se realiza una predicción de nuestras imágenes.

Esta predicción nos indica el porcentaje de buena y mala calidad que estas poseen, tal como lo podemos observar en el gráfico 2.13, nos muestra los granos de cacao en pulpa que nosotros vamos a testear, así como el resultado en el gráfico 2.14.

Figura 2.13: Imagen de prueba

```
La imagen es 99.99986410140991 % de Buena Calidad y un 0.00013589859008789062 % de Mala
Calidad
```

Figura 2.14: Porcentaje de buena y mala calidad de una imagen

3. RESULTADOS

3.1. Presentación y Discusión de Resultados

Para presentar nuestros datos y nuestros resultados, empezaremos con una pequeña muestra de nuestra base de datos. En la figura 3.1 podemos observar la imagen tal y como llego a nuestras manos en la base de datos de la Universidad de Quevedo.

Figura 3.1: Imagen muestra de nuestra base de datos

La imagen pasa por nuestro eliminador de fondo y obtenemos el resultado observado en la figura 3.2.

Figura 3.2: Imagen muestra de nuestra base de datos sin fondo

Luego de esto realizamos el proceso de tiling, este proceso me permite separar en pedazos

iguales nuestra imagen y obtenemos resultados como los de la figura 3.3.

Figura 3.3: Imagen muestra ya separada en pedazos pequeños “tiles”

En nuestro clasificador, cada tile es analizado y obtiene un porcentaje ya se de buena o de mala calidad, si el porcentaje de buena calidad es mayor al 50 % el tile se pinta de azul, caso contrario el tile se pinta de rojo y nos indica el porcentaje de buena calidad que dispone, eso lo observamos en la figura 3.4.

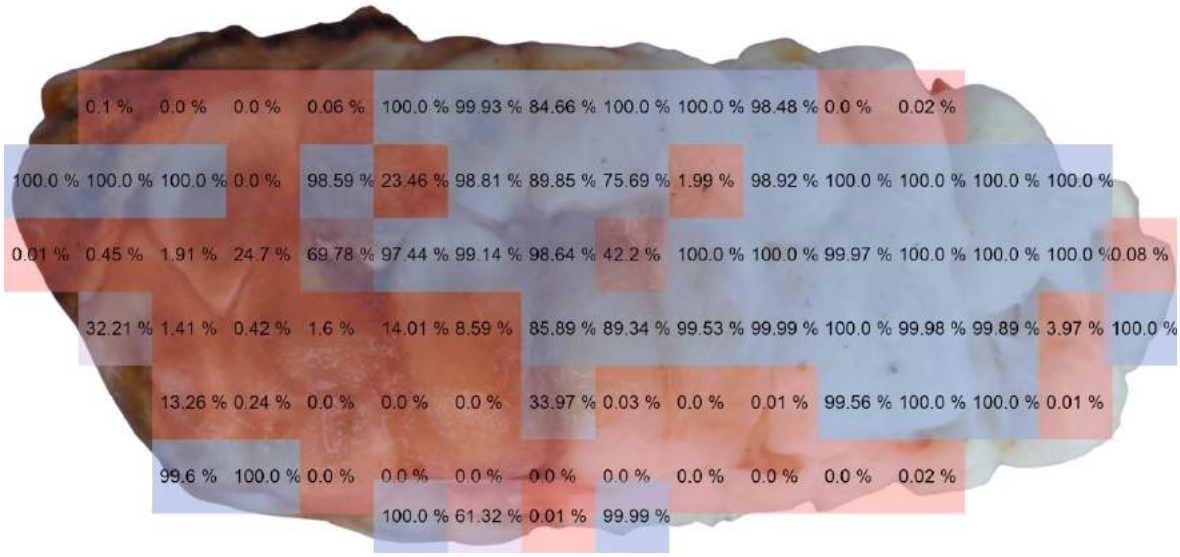


Figura 3.4: Cada segmento de imagen obtiene su respectiva ponderación.

Realizamos nuestro promedio con la fórmula 3.1.

$$Promedio = \frac{\sum_{i=1}^{NumerodeTiles} PorcentajeBuenacalidad(i)}{NumerodeTiles} \tag{3.1}$$

En el caso de esta imagen de Calidad Media, obtuvimos un 54.27 % de Buena Calidad y un 45.73 % de Mala Calidad, es decir esta en el rango preestablecido de Media Calidad y su afirmación es correcta.

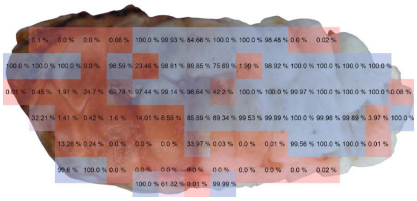
```
La imagen es 54.2718096199828 % de Buena Calidad y un 45.7281903800172 %
de Mala Calidad
```

Figura 3.5: Resultado en nuestro clasificador

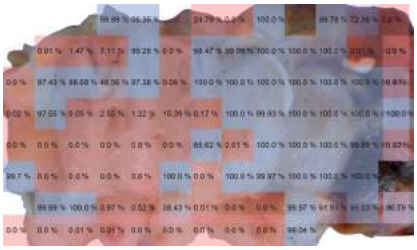
Resultados obtenidos al analizar las imágenes de Media Calidad

(a) Imagen de Media Calidad 1

(b) Imagen de Media Calidad 2

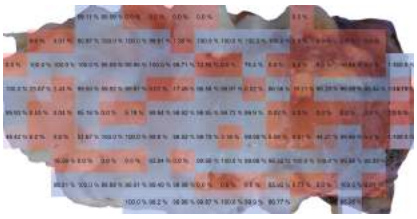


(c) Imagen de Media Calidad 3

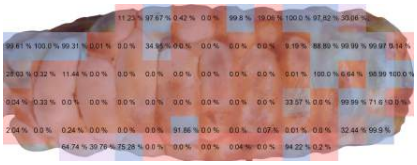


(d) Imagen de Media Calidad 4

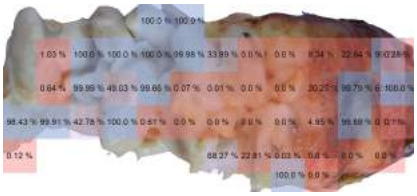
(e) Imagen de Media Calidad 5



(f) Imagen de Media Calidad 6



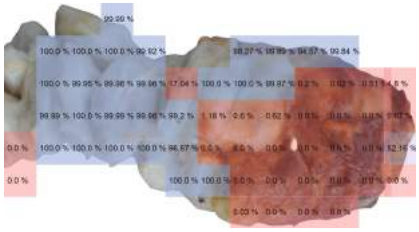
(g) Imagen de Media Calidad 7



(h) Imagen de Media Calidad 8

(i) Imagen de Media Calidad 9

(j) Imagen de Media Calidad 10



(k) Imagen de Media Calidad 11

Figura 3.6: Varias Imágenes y sus porcentajes de Buena Calidad por Tiles.

Como podemos observar en la figura 3.6 las 10 muestras de Media Calidad fueron ingresadas a nuestro clasificador, el cual, luego de todo el proceso citado anteriormente, pudo extraer las características de cada una de ellas y todas las predicciones fueron acertadas. Con estos resultados concluimos que, uno de los mas complejos de analizar para nuestra Red Neuronal Convolucional fue el de la figura 6(g), el cual podemos ver originalmente en 3.7; esta imagen al tener aproximadamente un 90 % de “propiedades intermedias”, es decir su color no es ni blanco como el de buena calidad ni marrón oscuro como el de mala calidad afecta al procesamiento correcto de los datos.

Figura 3.7: Imagen Original

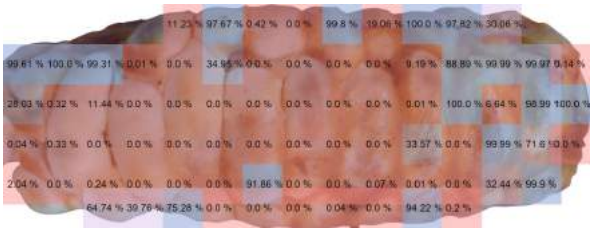


Figura 3.8: Imagen Clasificada

La clasificación realizada por la Red Neuronal Convolucional de granos de cacao en base a pulpa nos indica, aun con algunos inconvenientes, los siguientes datos:

El valor 24.8943065715743 % correspondiente a la Buena Calidad; y 75.1056934284257 % equivalente a la Mala Calidad, nos permite situarlo en un punto medio entre estas categorías, lo que dificulta la clasificación o no de los granos, incluso para el CNN.

Ayudándonos de la tabla 3.1 para analizar los datos de nuestro Cacao de Media Calidad podemos interpretar estos resultados como un acierto, ya que los posibles “errores”, que se presentaron en la red, están en función de la clasificación de los productores y agricultores y la base de datos que formen.

Imagen	% Buena Calidad	% Mala Calidad	Etiqueta
1	51.7409219450217	48.2590780549783	MEDIA CALIDAD
2	47.1702525062382	52.8297474937618	MEDIA CALIDAD
3	32.7405538048902	67.2594461951098	MEDIA CALIDAD
4	46.0158212375575	53.9841787624425	MEDIA CALIDAD
5	66.9628737968162	33.0371262031838	MEDIA CALIDAD
6	24.8943065715743	75.1056934284257	MEDIA CALIDAD
7	62.9705287502969	37.0294712497031	MEDIA CALIDAD
8	40.4122761277809	59.5877238722191	MEDIA CALIDAD
9	50.2840047030964	49.7159952969036	MEDIA CALIDAD
10	55.3984651169519	44.6015348830481	MEDIA CALIDAD
11	53.6919062697578	46.3080937302422	MEDIA CALIDAD
12	17.1780375175142	82.8219624824858	MEDIA CALIDAD
13	17.1104499847326	82.8895500152674	MEDIA CALIDAD
14	20.7420586787501	79.2579413212499	MEDIA CALIDAD
15	36.7181013137316	63.2818986862684	MEDIA CALIDAD
16	54.1044980496147	45.8955019503853	MEDIA CALIDAD
17	44.033818545361	55.966181454639	MEDIA CALIDAD
18	24.008100512606	75.991899487394	MEDIA CALIDAD
19	20.4707234617292	79.5292765382708	MEDIA CALIDAD
20	1.34580165671148	98.6541983432885	MEDIA CALIDAD
21	90.9757714649459	9.02422853505411	MEDIA CALIDAD
22	92.8894872367382	7.1105127632618	MEDIA CALIDAD
23	50.0298473483372	49.9701526516628	MEDIA CALIDAD
24	53.0921937996704	46.9078062003296	MEDIA CALIDAD
25	70.1084179270812	29.8915820729188	MEDIA CALIDAD
26	50.2495305492808	49.7504694507192	MEDIA CALIDAD
27	28.4554431391052	71.5445568608948	MEDIA CALIDAD
28	37.902404525915	62.097595474085	MEDIA CALIDAD
29	40.0892956091239	59.9107043908761	MEDIA CALIDAD
30	70.7960252816984	29.2039747183016	MEDIA CALIDAD
31	56.2880012545931	43.7119987454069	MEDIA CALIDAD
32	32.310779697401	67.689220302599	MEDIA CALIDAD
33	38.7925196578765	61.2074803421235	MEDIA CALIDAD
34	41.5626729207536	58.4373270792464	MEDIA CALIDAD
35	22.4169587849272	77.5830412150728	MEDIA CALIDAD
36	27.3191212733276	72.6808787266724	MEDIA CALIDAD
37	2.69761829763166	97.3023817023683	MEDIA CALIDAD
38	53.8463288012211	46.1536711987789	MEDIA CALIDAD
39	8.56891377262645	91.4310862273736	MEDIA CALIDAD
40	68.2111866098627	31.7888133901373	MEDIA CALIDAD
41	8.20244597669626	91.7975540233038	MEDIA CALIDAD
42	73.8461808577521	26.1538191422479	MEDIA CALIDAD
43	74.7039634653944	25.2960365346056	MEDIA CALIDAD
44	40.5813639697671	59.4186360302329	MEDIA CALIDAD
45	30.7446529869821	69.2553470130179	MEDIA CALIDAD
46	36.1041006185982	63.8958993814018	MEDIA CALIDAD
47	64.606495603886	35.393504396114	MEDIA CALIDAD
48	74.3151081536187	25.6848918463813	MEDIA CALIDAD
49	83.5928902047647	16.4071097952353	MEDIA CALIDAD
50	84.4445839353456	15.5554160646544	MEDIA CALIDAD
51	70.6620968630905	29.3379031369095	MEDIA CALIDAD
52	75.845073667715	24.154926332285	MEDIA CALIDAD
53	65.5031680103894	34.4968319896106	MEDIA CALIDAD
54	91.6073662401697	8.3926337598303	MEDIA CALIDAD
55	45.0829371420316	54.9170628579684	MEDIA CALIDAD
56	88.3470973983597	11.6529026016403	MEDIA CALIDAD
57	82.8891064888368	17.1108935111632	MEDIA CALIDAD
58	86.7510032772813	13.2489967227187	MEDIA CALIDAD
59	55.0981880534583	44.9018119465417	MEDIA CALIDAD
60	47.0259569560131	52.9740430439869	MEDIA CALIDAD
61	65.8956442737055	34.1043557262945	MEDIA CALIDAD
62	54.2718096199828	45.7281903800172	MEDIA CALIDAD

Tabla 3.1: Tabla de Resultados en nuestras Imágenes de Media Calidad

Dos claros ejemplos antagónicos son las Imágenes #20 9(a) y #21 9(b), cada una de ellas fue clasificada en:

- Imagen #20: 1.35 % de Buena Calidad y 98.65 % de Mala Calidad.**Mala Calidad**
- Imagen #21: 90.98 % de Buena Calidad y 9.02 % de Mala Calidad. **Buena Calidad**

Estos datos podrían parecer simples errores, pero si nos ponemos a analizar una red neuronal y entendemos como extrae las características de cada imagen, para después actuar como un clasificador; Y a su vez analizamos las imágenes, encontramos que efectivamente estos no son errores de nuestra Red Neuronal.

(a) Imagen Clasificada #20 (b) Imagen Clasificada #21

Figura 3.9: Imágenes Clasificadas en Media Calidad.

Como podemos observar, estos resultados no están para nada lejos de la realidad, ya que al nosotros interpretar los resultados, la imagen 9(b) en efecto tiene un porcentaje aproximado al 90 % de Buena Calidad, y contrario a esto, la imagen 9(a) posee mas del 90 % de su pulpa en mal estado con colores opacos y presenta putrefacciones en partes del centro.

Resultados Generales

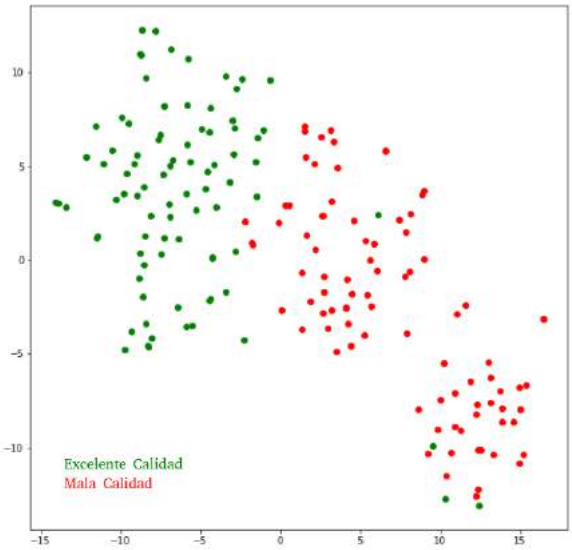


Figura 3.10: Gráfico t-SNE del clasificador

La figura 3.10 muestra el t-SNE (T-distributed Stochastic Neighbor Embedding) de nuestra propuesta. En este gráfico podemos apreciar que las muestras se pueden separar aproximadamente de forma lineal, lo que justifica nuestras elecciones al momento de entrenar y validar nuestro modelo. Este modelo nos permite observar, que existieron únicamente 4 errores:

- 1 falso negativo
- 3 falsos positivos

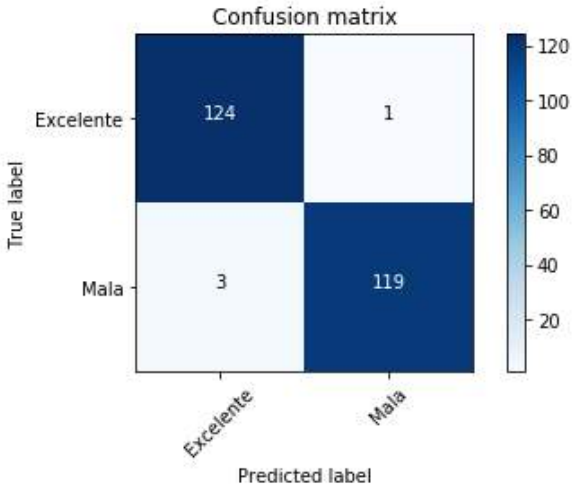


Figura 3.11: Matriz de Confusión del Clasificador

Esto lo podemos confirmar con la ayuda de la Matriz de confusión que realizamos en la figura 3.11, en esta observamos que 3 imágenes que en realidad eran de Mala Calidad fueron clasificadas como de Buena Calidad, y una imagen que era de Buena Calidad fue clasificada como de Mala Calidad.

$$\text{Exactitud} = \frac{VP + VN}{P + N} \tag{3.2}$$

$$\text{Precisión} = \frac{VP}{VP + FP} \tag{3.3}$$

$$\text{Exhaustividad} = \frac{VP}{P} \tag{3.4}$$

$$\text{Especificidad} = \frac{VN}{N} \tag{3.5}$$

$$\text{Valor-F} = \frac{2(\text{Precisión} \cdot \text{Exhaustividad})}{\text{Precisión} + \text{Exhaustividad}} \tag{3.6}$$

Utilizamos las fórmulas 3.2 3.3 3.4 3.5 3.6 para calcular la Exactitud, Exhaustividad, Especificidad, Precisión y ValorF de nuestro clasificador.

Exactitud	98.38 %
Precisión	97.64 %
Exhaustividad	99.20 %
Especificidad	97.54 %
Valor-F	98.41 %

Tabla 3.2: Resultados para evaluar el rendimiento de la Red Neuronal Convolucional

Al encontrar los resultados de la tabla 3.2 La exactitud nos indica que un 98.38 % de las imágenes fue clasificado correctamente, este dato no es muy importante para analizar los resultados de nuestra red neuronal, pero ya nos da buenos niveles que nos dan una forma general de como esta nuestro proyecto. La precisión que tiene un valor de 97.64 % nos indica el porcentaje de Buena Calidad que esta bien clasificado, la exhaustividad o sensibilidad con un valor de 99.20 % nos indica el porcentaje de Buena calidad que fue clasificado del total de muestras que son de Buena Calidad en realidad y nos indica que nuestro sistema tiene bastante sensibilidad para no afectar a la clase que es mas importante para nosotros que es la buena calidad, ya que esta es la que necesitamos saber para mejorar la industria;

La especificidad nos muestra cual es el porcentaje de todas las muestras que son de Mala Calidad que fue correctamente clasificado y finalmente realizamos una prueba de Valor-F que, aunque no sea necesaria al indicarnos que posee un 98.41 %, al existir un desequilibrio entre clases en la vida real es mejor utilizar este valor, ya que nos da una noción mas verdadero que la exactitud, y en nuestro caso es un poco mayor.

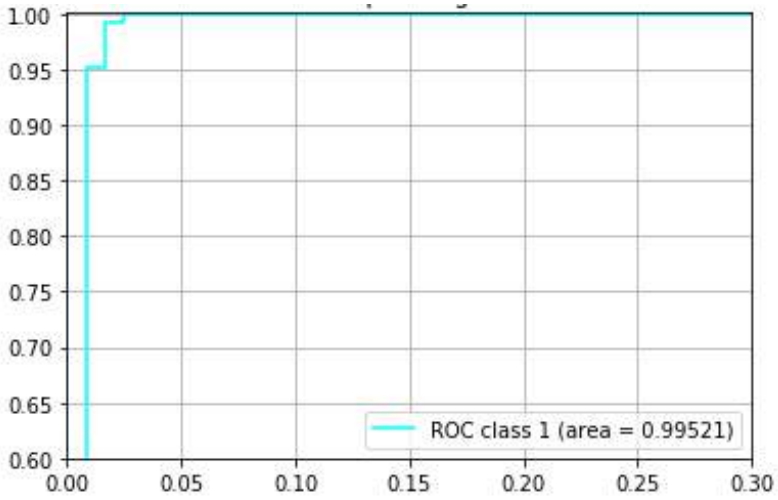
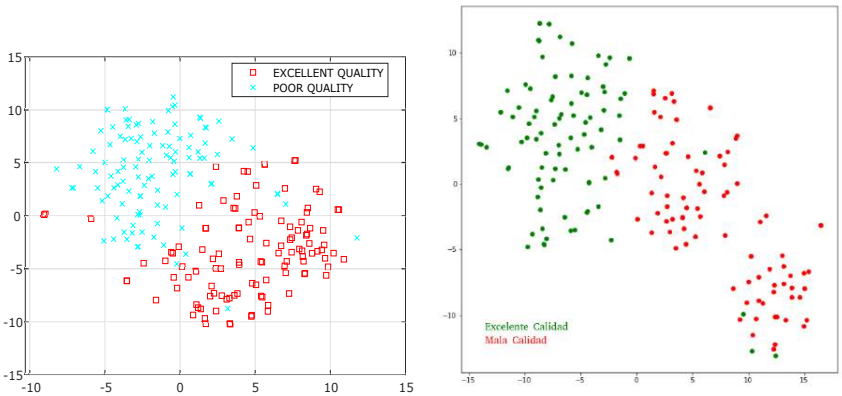


Figura 3.12: Curva ROC y Área Bajo la Curva

Con estos resultados, realizamos nuestro curva ROC 3.12 que nos muestra que el rendimiento de nuestro modelo de clasificación es muy bueno en todos los umbrales de clasificación ya que el área bajo la curva calculada nos dio un valor de 0.99521, muy cercano a 1, lo que nos indica que los errores al clasificar con nuestro sistema serán mínimos.

Resultados de la comparación con el proyecto basado en una máquina de vectores de soporte (SVM)

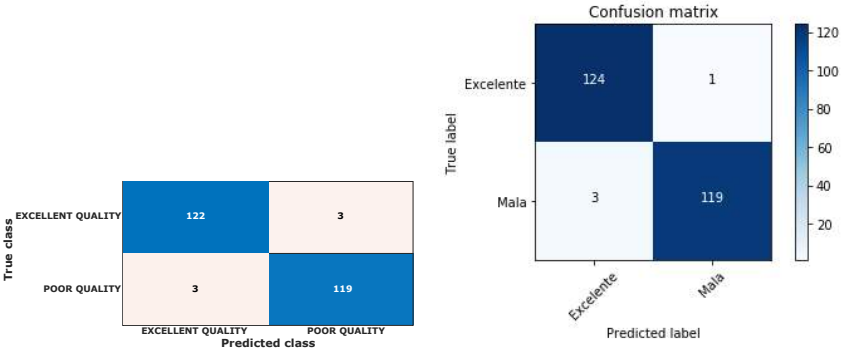
Cuando se realiza una comparación entre la Red Neuronal Convolucional y el proyecto que tiene sus bases en la maquina de vectores de soporte (SVM) se puede reconocer que el tipo de clasificación usado en el segundo se limita, única y exclusivamente, a 2 clases. Por el contrario, el CNN realiza un encasillamiento de tres clases lo cual ha permitido que alcance una tasa de exactitud de hasta un 95 %; representado como un programa de clasificación confiable y capaz de producir los resultados esperados para las organizaciones. Sin embargo, es importante que la comparación se la realice tomando en cuenta los mismo parámetros de evaluación: dos clases.



(a) Gráfico t-SNE SVM (b) Gráfico t-SNE CNN

Figura 3.13: Gráficos t-SNE de los dos proyectos

Podemos observar en el gráfico t-SNE que la tasa de errores es mucho mas baja que la del proyecto basado en SVM. Los datos se encuentran dispersos en la figura 13(a), a diferencia de la Red Neuronal Convolucional, que presenta datos de manera compacta, haciendo que la tasa de error sea mínima.



(a) Matriz de Confusion SVM (b) Matriz de Confusion CNN

Figura 3.14: Imágenes Clasificadas en Media Calidad.

Al observar las matrices de confusión en la figura 3.14 se puede identificar que los falsos negativos del proyecto SVM es mayor al de la Red Neuronal Convolucional; teniendo en el primero un valor de 3, dos puntos mas que el CNN. Los falsos negativos (1) que existen en nuestro proyecto evidencian una tasa de error aceptable.

Tabla 3.3: Valores obtenidos desde las matrices de confusión

	SVM	CNN
Exactitud	97.57 %	98.38 %
Precisión	97.6 %	97.64 %
Exhaustividad	97.6 %	99.2 %
Especificidad	97.54 %	97.54 %
f1-score	97.6 %	98.41 %

Según la tabla 3.3 todos y cada uno de los valores obtenidos en el proyecto: la exactitud, la precisión, el valor f, la especificidad y la exhaustividad ubicados en la tabla 3.3 evidencian que la Red Neuronal Convolucional tiene un rendimiento superior al proyecto que se basa en una maquina de vectores de soporte. Los resultados son los esperados para el CNN.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

La transferencia de aprendizaje llevada a cabo en el proyecto nos permite llegar a la conclusión de que las fases para el tratamiento de cacao a través de la CNN son necesarias: realizando, en primer momento, una especificación general a través de una red neuronal pre entrenada (VGG19) con millones de imágenes de la base datos de ImageNet y, en un segundo momento, una extracción de características de nuestra fuente de imágenes que evite errores futuros en su aplicación. Las fases de entrenamiento, validación y prueba explican su perfeccionamiento y la baja tasa de error que tiene el programa.

Mediante la realización de este proyecto se ha podido concluir que el problema que gira en torno a la clasificación, procesamiento y tratamiento de la pulpa de cacao a través de cultivadores y productores ha hecho posible proponer el desarrollo de una Red Neuronal Convolucional enfocada a mejorar los tiempos de encasillamiento de los granos de cacao en: Excelente calidad, media calidad y mala calidad.

La aplicación de una Red Neuronal Convolucional para el tratamiento de cacao, en comparación con la clasificación tradicional a cargo de los cultivadores y pequeños productores, permite un salto relevante que masifica el encasillamiento de los granos de cacao. Estas acciones hacen posible la obtención de granos “Nacionales” de muy buena calidad al mismo tiempo que se desecha a aquellos que no cumplen con los estándares propuestos para su procesamiento.

De igual manera, es importante reconocer que nuestra Red Neuronal Convolucional no se destaca por ser un clasificador en tres categorías (excelente calidad, media calidad y mala calidad) debido a la superficialidad de su medición. Como respuesta se realizo un sistema de promediado de dos categorías; método que permitió obtener resultados mas exactos y precisos en torno a la clasificación de los granos de cacao en pulpa.

Finalmente se puede concluir que los valores de precisión (99.2 %), especificación (97.54 %), exactitud (98.3 %) y el área bajo la curva (99.52 %) son ligeramente superiores en comparación al sistema basado en una máquina de vectores de soporte (SVM) demostrando que nuestra red neuronal basada en la transferencia de aprendizaje es prometedora en la industria cacaotera y facilita técnicas que no han sido usadas por los productores ni agricultores con anterioridad. Ademas, el sistema fue estructurado para no representar dificultades a la hora de implementarlo ya que toda la programación fue simplificada; para que no se deman-

de un gran poder computacional e inversiones económicas grandes para su aplicación.

4.2. RECOMENDACIONES

Se obtuvieron buenos resultados al implementar una Red Neuronal Convolucional de clasificación de granos de cacao en base a su pulpa, por esa razón, es necesario realizar algunas recomendaciones las cuales permitan aprovechar al máximo este proyecto y reducir el margen de error. Es importante que se tome en cuenta para futuros proyectos que, para una medición mas exacta, se debe realizar una red neuronal de no mas de dos categorías las cuales nos permitan situar características generales, las que a su vez facilitaran la apertura de subcategorías.

Otra recomendación que es pertinente realizar es ajustar la actualización de base de datos y contrastar con nuevas fuentes nacionales e internacionales; factores que permitan un tratamiento más completo, lleno de argumentos y aprendizajes que hagan de la clasificación cacaotera un tema de importancia mundial en beneficio de las grandes empresas así como de los millones de consumidores de cacao en el mundo; permitiéndoles una experiencia diferente.

Es importante, como recomendación para quienes deseen poner en practica este proyecto, que se preste total atención al código de programación utilizado, ya que este se encuentra limpio, claro, ordenado y con anotaciones lo que permite que sea utilizado y modificado para cubrir las necesidades de proyectos que en el futuro necesiten de un clasificador con excelentes características.

De igual manera, es necesario que antes de empezar con la programación de la Red Neuronal Convolucional se tenga conocimientos de transferencia de aprendizaje, aprendizaje profundo y aprendizaje de maquina, debido a que su comprensión y entendimiento nos permitirá determinar de manera clara los objetivos del proyecto que vayamos a trazar, así como también las metodologías y las herramientas que harán posible su concreción.

En caso de aplicar nuestra Red Neuronal Convolucional de clasificación de granos de cacao con pulpa es recomendable realizar evaluaciones constantes capaces de detectar anomalías en su aplicación o posibles errores que puedan aparecer. Un cronograma de valoración es una herramienta importante que se utilizara para monitorear y conseguir los resultados de exactitud y clasificación deseada.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] J. Freire, *La comercializacion del cacao*, 1st ed. CAMAREN, 2009, vol. 1.

[2] M. Diosdado Contreras and E. Méndez Bolaina, “Cacao: Manjar de dioses, elixir de vida para los mortales,” *Luz y Vida*, 2018.

[3] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.

[4] R. Yamashita, M. Nishio, R. K. Do, and K. Togashi, “Convolutional neural networks: An overview and application in radiology,” *Insights into Imaging*, vol. 9, no. 4, p. 611–629, 2018.

[5] J. Quiroz and F. Amores, “Rehabilitación de plantaciones tradicionales de cacao en ecuador,” *CATIE*, vol. 63, p. 73–80, 2013.

[6] M. E. Albarracín Macías and K. D. Moposita Ortega, “Diseño y simulación de una máquina clasificadora por visión artificial y despulpadora de cacao,” B.S. thesis, Quevedo-UTEQ, 2018.

[7] T. Himblot, “Data augmentation : Boost your image dataset with few lines of python,” Mar 2018. [Online]. Disponible en: https://medium.com/@thimblot/data-augmentation-boost-your-image-dataset-with-few-lines-of-python-155c2dc1baec

[8] A. Jay, “What is matlab function & matlab plot?” 2022. [Online]. Disponible en: https://cydomedia.com/what-is-matlab/

[9] E. Shein, “Python for beginners,” *Communications of the ACM*, vol. 58, no. 3, 2015.

[10] E. Witman, “What is python? the popular, scalable programming language, explained,” 2022. [Online]. Disponible en: https://www.businessinsider.com/what-is-python

[11] O. Bar-el, “Getting the most out of your google colab,” 2021. [Online]. Disponible en: https://medium.com/@oribarel/getting-the-most-out-of-your-google-colab-2b0585f82403

[12] I. Center, “Tpu vs gpu: Pros and cons,” 2022. [Online]. Disponible en: https://www.inmotionhosting.com/support/product-guides/private-cloud/tpu-vs-gpu

[13] S. Villagómez García and F. Argüello Moreta, “OptimizaciÓn y aprovechamiento del residuo (exudado del mucÍlago) de la almendra fresca del cacao (theobroma cacao l.) ccn51 en la elaboraciÓn de vinagre,” *Tsafiqui - Revista Científica en Ciencias Sociales*, no. 4, pp. 7–19, 2013.

[14] H. Aguilar, *Manual para la Evaluación de la Calidad del Grano de Cacao*, 1st ed. FUNDACIÓN HONDUREÑA DE INVESTIGACIÓN AGRÍCOLA, 2016.

[15] J. Brownlee, “A gentle introduction to the imagenet challenge (ilsvrc),” 2022. [Online]. Disponible en: https://machinelearningmastery.com/introduction-to-the-imagenet-large-scale-visual-recognition-challenge-ilsvrc

[16] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014. [Online]. Disponible en: https://arxiv.org/abs/1412.6980

[17] K. Team, “Keras documentation: Adam,” 2022. [Online]. Disponible en: https://keras.io/api/optimizers/adam/

[18] 2022. [Online]. Disponible en: https://www.tensorflow.org/api_docs/python/tf/keras/losses/CategoricalCrossentropy

[19] 2022. [Online]. Disponible en: https://www.aprendemachinelearning.com/setsdeentrenamientotestvalidacioncruzada

[20] 2022. [Online]. Disponible en: https://www.tensorflow.org/api_docs/python/tf/keras/Model

[21] M. Bertin, *Training, Evaluating, and Tuning Deep Neural Network Models with TensorFlow-Slim: Advanced Topics in Training, Evaluating, and Tuning Deep Neural Network Models*. O’Reilly, 2017. [Online]. Disponible en: https://books.google.com.ec/books?id=S3s7zQEACAAJ

ANEXOS

Los siguientes anexos se encuentran en formato digital:

ANEXO A. TransferLearningPrueba3.py

ANEXO B. imagesplitcoords.py

ANEXO C. PruebaResultados.py

ANEXO D. imagesplitcoords.py

ANEXO E. Unir.py