,

UNIVERSIDAD ESTATAL DE MILAGRO

FACULTAD CIENCIAS E INGENIERÍA

INFORME DE PROYECTO INTEGRADOR

PREVIO A LA OBTENCIÓN DEL TÍTULO DE **INGENIERO(A) EN SISTEMAS COMPUTACIONALES**

TEMA: DESARROLLO DE UNA RED NEURONAL CONVOLUCIONAL PARA LA CLASIFICACIÓN DE BOTELLAS PLÁSTICAS EN UNA EMPRESA ENVASADORA DE BEBIDAS GASEOSAS DE LA CIUDAD DE MILAGRO

Autores:

Srta. LUNA BUSTAMANTE GENESIS JANETH

Sr. VASCONEZ CAMINO KLEYNER ARIEL

Tutor:

Mgtr. ORTIZ MATA JHONNY DARWIN

Milagro, Febrero 2020

ECUADOR

DERECHOS DE AUTOR

Ingeniero.
Fabricio Guevara Viejó, PhD.
RECTOR
Universidad Estatal de Milagro
Presente.

Yo, LUNA BUSTAMANTE GENESIS JANETH, en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de integración curricular, modalidad presencial, mediante el presente documento, libre y voluntariamente procedo a hacer entrega de la Cesión de Derecho del Autor, como requisito previo para la obtención de mi Título de Grado, como aporte a la Línea de Investigación tecnologías de la información y de la comunicación, de conformidad con el Art. 114 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación, concedo a favor de la Universidad Estatal de Milagro una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Conservo a mi favor todos los derechos de autor sobre la obra, establecidos en la normativa citada.

Así mismo, autorizo a la Universidad Estatal de Milagro para que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

El autor declara que la obra objeto de la presente autorización es original en su forma de expresión y no infringe el derecho de autor de terceros, asumiendo la responsabilidad por cualquier reclamación que pudiera presentarse por esta causa y liberando a la Universidad de toda responsabilidad.

Milagro, 17 de febrero de 2020

Génesis Luna B.

LUNA BUSTAMANTE GENESIS JANETH
Autor 1
CI: 0942100777

DERECHOS DE AUTOR

Ingeniero.
Fabricio Guevara Viejó, PhD.
RECTOR
Universidad Estatal de Milagro
Presente.

Yo, VASCONEZ CAMINO KLEYNER ARIEL, en calidad de autor y titular de los derechos morales y patrimoniales del trabajo de integración curricular, modalidad matutina, mediante el presente documento, libre y voluntariamente procedo a hacer entrega de la Cesión de Derecho del Autor, como requisito previo para la obtención de mi Título de Grado, como aporte a la Línea de Investigación tecnologías de la información y de la comunicación, de conformidad con el Art. 114 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación, concedo a favor de la Universidad Estatal de Milagro una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos. Conservo a mi favor todos los derechos de autor sobre la obra, establecidos en la normativa citada.

Así mismo, autorizo a la Universidad Estatal de Milagro para que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

El autor declara que la obra objeto de la presente autorización es original en su forma de expresión y no infringe el derecho de autor de terceros, asumiendo la responsabilidad por cualquier reclamación que pudiera presentarse por esta causa y liberando a la Universidad de toda responsabilidad.

Milagro, 17 de febrero de 2020

Kley Vasconez

VASCONEZ CAMINO KLEYNER ARIEL
Autor 2
CI: 0920993441

APROBACIÓN DEL TUTOR DEL TRABAJO DE INTEGRACIÓN CURRICULAR

Yo, ORTIZ MATA JHONNY DARWIN en mi calidad de tutor del trabajo de integración curricular, elaborado por los estudiantes LUNA BUSTAMANTE GENESIS JANETH y VASCONEZ CAMINO KLEYNER ARIEL, cuyo título es Desarrollo de una Red Neuronal Convolucional para la Clasificación de Botellas Plásticas en una Empresa Envasadora de Bebidas Gaseosas de la Ciudad de Milagro, que aporta a la Línea de Investigación tecnologías de la información y de la comunicación previo a la obtención del Título de Grado, Ingeniero en Sistemas Computacionales; considero que el mismo reúne los requisitos y méritos necesarios en el campo metodológico y epistemológico, para ser sometido a la evaluación por parte del tribunal calificador que se designe, por lo que lo APRUEBO, a fin de que el trabajo sea habilitado para continuar con el proceso previa titulación de la alternativa de Trabajo de Integración Curricular de la Universidad Estatal de Milagro.

Milagro, 17 de febrero de 2020

ORTIZ MATA JHONNY DARWIN

Tutor
C.I: 0927159111

APROBACIÓN DEL TRIBUNAL CALIFICADOR

El tribunal calificador constituido por:

Mgtr. Ortiz Mata Jhonny Darwin

Mgtr. Mendoza Cabrera Denis Dario

Mgtr. Arévalo Gamboa Lissett Margarita

Luego de realizar la revisión del Trabajo de Integración Curricular, previo a la obtención del título (o grado académico) de INGENIERA EN SISTEMAS COMPUTACIONALES presentado por la estudiante Luna Bustamante Genesis Janeth

Con el tema de trabajo de Titulación: Desarrollo de una Red Neuronal Convolucional para la Clasificación de Botellas Plásticas en una Empresa Envasadora de Bebidas Gaseosas de la Ciudad de Milagro.

Otorga al presente Trabajo de Integración Curricular, las siguientes calificaciones:

Trabajo Integración Curricular	[59.33]
Defensa oral	[39.33]
Total	[98.66]

Emite el siguiente veredicto: (aprobado/reprobado) APROBADO

Fecha: 17 de febrero de 2020

Para constancia de lo actuado firman:

	Nombres y Apellidos	Firma
Presidente	Ortiz Mata Jhonny Darwin	
Secretario /a	Mendoza Cabrera Denis Dario	
Integrante	Arévalo Gamboa Lissett Margarita	

APROBACIÓN DEL TRIBUNAL CALIFICADOR

El tribunal calificador constituido por:

Mgtr. Ortiz Mata Jhonny Darwin

Mgtr. Mendoza Cabrera Denis Dario

Mgtr. Arévalo Gamboa Lissett Margarita

Luego de realizar la revisión del Trabajo de Integración Curricular, previo a la obtención del título (o grado académico) de INGENIERO EN SISTEMAS COMPUTACIONALES presentado por el estudiante Vásconez Camino Kleyner Arel

Con el tema de trabajo de Titulación: Desarrollo de una Red Neuronal Convolucional para la Clasificación de Botellas Plásticas en una Empresa Envasadora de Bebidas Gaseosas de la Ciudad de Milagro.

Otorga al presente Proyecto Integrador, las siguientes calificaciones:

Trabajo de Integración Curricular	[59.33]
Defensa oral	[39]
Total	[98.33]

Emite el siguiente veredicto: (aprobado/reprobado) APROBADO

Fecha: 17 de febrero de 2020

Para constancia de lo actuado firman:

	Nombres y Apellidos	Firma
Presidente	Ortiz Mata Jhonny Darwin	
Secretario /a	Mendoza Cabrera Denis Dario	
Integrante	Arévalo Gamboa Lissett Margarita	

DEDICATORIA

Dedico esto a mis padres Lenin y Flor por su un ejemplo a seguir y mi apoyo incondicional, también a mis hermanos Daniela y Lenin para que les sirva de referencia de lo que es la perseverancia y dedicación.

DEDICATORIA

A mis padres Amparito y Sabino por su paciencia y sacrificios dados para llegar a ser lo que me proponga.

A mis hermanos Edson y Jair los cuales me han servido como ejemplo de inspiración y ejemplo a seguir ante cualquier adversidad. Sus maneras de ser hacen que ante cualquier problema la paciencia sea lo primordial.

AGRADECIMIENTO

Mi primer agradecimiento es para Dios porque me dio la fuerza y voluntad de seguir adelante en este camino que muchas veces fue difícil.

Gracias a mis padres por ser mi gran fuente de apoyo, que gracias a su esfuerzo he logrado una meta más, también a mis hermanos que me acompañaron desde un principio también incluyo a mi Tía Miriam por ser parte de este proceso porque no solo me apoyó a mí, sino también ayudó a mis papás en esta largo labor y por último a mi abuelita por estar pendiente de mi bienestar y salud.

A mi amiga de toda la vida Lissette quien desde lejos me motiva a seguir adelante y a cumplir mis metas. A mi compañero de tesis y novio Kleyner con quien trabajamos horas y días en este proyecto de investigación que es de gran importancia, convirtiéndose en aquella persona que siempre estaba dispuesto a ayudar.

Igualmente, a mi tutor Ing. Jhonny Ortiz por ser el guía en este proceso, por estar pendiente de cada etapa y por verificar cada mínimo detalle del proyecto.

AGRADECIMIENTO

Primero a Dios por permitirme vivir y conocer cosas interesantes del mundo y su ciencia creada a través de los años.

A mis padres y toda mi familia por sus valores y virtudes dadas que han hecho de mí, la persona que alegre y feliz que soy.

A mí compañera de tesis y novia Génesis por su paciencia y apoyo incondicional al momento de redactar juntos por largas horas y días el presente proyecto de investigación y moldear mis ideas. Sus ánimos, cariños y momentos especiales han hecho de la investigación algo mucho más ligero de llevar dándose situaciones divertidas y alegres. A mis amigos Christian, Andy y demás compañeros cercanos con los he tenido la oportunidad de impartir y adquirir conocimientos.

A Newton, Curie, Tesla entre otros científicos y influencers que han dejado su legado en múltiples avances entorno a la ciencia que han me han permitido tener el mismo espíritu investigador.

A docentes de la Universidad Estatal de Milagro como lo es mí tutor Ing. Jhonny Ortiz que ha estado en constante revisión de cada parte del documento. Para él y los demás docentes estoy agradecido ya que a su manera me han logrado impartirme sus conocimientos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Título de Trabajo Integración Curricular: Desarrollo de una Red Neuronal Convolucional para la Clasificación De Botellas Plásticas en una Empresa Envasadora de Bebidas Gaseosas de la Ciudad De Milagro.

RESUMEN

La creación de las redes neuronales ha generado grandes avances en diversos campos científicos como la medicina, la industria, la tecnología entre otros. Causando en ellos una mejora en su rendimiento y de esta manera facilitando el funcionamiento a los usuarios. El presente proyecto de investigación se desarrolló una red neuronal convolucional que permitió el reconocimiento de botellas plásticas, para luego clasificarlas, con la utilización del lenguaje de programación Python y sus librerías aptas para la formación de una red neuronal, además se requirió del manejo de una cámara web en el que se pueda visualizar las botellas a examinar en tiempo real. En la ejecución del aplicativo se observó como resultado la efectividad de la red al momento de analizar un objeto a través de un prototipo, esta reconoce el tipo marca de las botellas de las categorías con las que se trabajó en el entrenamiento como lo es CocaCola® con un porcentaje de 97%, Sprite® se obtuvo un 85% y con menor valor de 72% Fanta®.

PALABRAS CLAVE: Reconocimiento, imágenes, red neuronal convolucional, Python.

Título de Trabajo Integración Curricular: Development of a Neuronal Convolutional Network for the Classification of Plastic Bottles in a Carbonated Beverage Packaging Company in the City of Miracle.

ABSTRACT

The creation of neural networks has generated great advances in various scientific fields such as medicine, industry, technology and others. It causes in them an improvement in their performance and thus facilitating the operation to the users. The present research project developed a convolutional neural network that allowed the recognition of plastic bottles, to later classify them, with the use of Python programming language and its libraries suitable for the formation of a neural network, also required the use of a webcam in which you can view the bottles to be examined in real time. In the execution of the application it was observed as a result the effectiveness of the network at the moment of analyzing an object through a prototype, this one recognizes the type mark of the bottles of the categories with which it was worked in the training as it is CocaCola® with a percentage of 97%, Sprite® was obtained 85% and with smaller value of 72% Fanta®.

KEY WORDS: Recognition, Images, Convolutional Neural Network, Python

CAPÍTULO 1

1. INTRODUCCIÓN

El ser humano desde su prehistoria siempre ha estado en búsqueda de la automatización, desde encontrar la mejor manera para cultivar, cazar o sociabilizar, pasando por múltiples etapas ya sean guerras o tratados históricos. Hasta la actualidad todos están en constante mejora comenzando con un boom como es el alunizaje, con soluciones a problemas globales como el internet, que ahora se encuentra casi a la altura de una necesidad básica en países desarrollados. Otro tipo de mejora se da por la creación de robots que realizan necesidades básicas, manejado por redes neuronales las cuales son capaces de adaptarse a cualquier tipo de problema, estas se encuentran en constante mejora y forman un nuevo campo de estudio. Las redes neuronales, son una parte esencial en el campo de la optimización y control de procesos, así como la inteligencia artificial que ha permitido desarrollar sistemas con un buen funcionamiento. En este proyecto se definirá conceptos básicos relacionados las redes neuronales convolucionales, además se describirá como esta metodología o herramientas, formar parte de nuevas tecnologías de producción.

El proyecto técnico está conformado por tres capítulos donde se documentan los conceptos básicos a utilizar, los procedimientos que se aplicaron y los resultados obtenidos del sistema creado para el reconocimiento y clasificación de botellas basados en el análisis de las etiquetas.

En el primer capítulo se detalla la problemática existente en los procesos de clasificación, el objetivo general del proyecto, los objetivos específicos en donde se describen las metas a corto plazo a cumplir, además puntualiza las razones por las cuales se realiza este proyecto dentro de la justificación, en el marco teórico se describen estudios relacionados con las redes neuronales en el campo de la automatización de procesos realizados en trabajos previos.

En el segundo capítulo se explican las metodologías utilizadas, la estructura que tiene el sistema, la arquitectura de la red neuronal convolucional, como se recolectaron los datos para realizar el debido análisis, las herramientas, los programas y las librerías aplicadas para la creación del aplicativo.

El último capítulo se presenta los resultados obtenidos luego del desarrollo y ejecución del sistema de reconocimiento de las marcas en las botellas con sus debidas pruebas, para

después emplear su debida clasificación y se detalla el funcionamiento de la herramienta a través de un manual.

1.1. Planteamiento del problema

Las empresas de bebidas gaseosas luego del proceso de envasado requieren que las botellas sean clasificadas de forma automática por categorías para posterior despacho, en la mayoría de casos dicha clasificación se realiza de forma manual. El volumen de producción es considerablemente alto comparado con otro tipo de fábricas, clasificar toda esa cantidad de productos toma mucho tiempo con el uso los métodos clásicos, a largo plazo estos se vuelven obsoletos, sin embargo, para el fabricante estos mecanismos son funcionales, causando que no tenga interés en adquirir nuevas tecnologías que permitan realizar la clasificación de forma óptima.

El desconocimiento de información sobre el uso de metodologías y técnicas en áreas de automatización que tienen buenas características, herramientas y beneficios aplicados en el proceso de identificación y clasificación de botellas dentro de las fábricas que realizan este tipo de función, se da porque existe una gran cantidad de empresas envasadoras que, mediante el uso de máquinas grandes, costosas y complejas, se encargan de clasificar y ordenar los envases según su contenido o función, es decir, hacen el uso de técnicas tradicionales para realizar el proceso de categorización.

Cuando una empresa desea innovar en el área de procesamientos, donde se pretende alcanzar la automatización con el uso de nuevas tecnológicas, empleando diversas técnicas y maniobras, esto en ocasiones genera un problema, debido a que se requiere de la contratación de una empresa externa, capaz de realizar estos tipos procesos y las capacitaciones a los empleados se vuelve algo agotador y tedioso, por lo que la mayoría de empresas se rehúsan al uso de innovación o renovación de actividades que se pueden desarrollar fácilmente, se ahorrarían recursos por tiempos de operación al automatizar los procesos.

1.2. Objetivos

1.2.1. Objetivo General

Desarrollar un esquema de red neuronal convolucional en Phyton, aplicado a un prototipo de clasificación de botellas plásticas mediante su etiqueta y color.

1.2.2. Objetivos Específicos

- Recolectar información para la construcción de la base de datos.
- Definir los recursos que se utilizarán para la programación de la red neuronal artificial.
- Enlazar la comunicación entre el software y hardware
- Configurar los parámetros de la red neuronal convolucional en Python
- Realizar pruebas de funcionamiento de la red en el prototipo.
- Generar un instructivo del uso de la herramienta.

1.3. Justificación

La herramienta que incorporará el uso de nuevas metodologías y técnicas, sirve para la optimización de procesos de clasificación de botellas, causando un mejor rendimiento mediante la reducción de los tiempos de operación, el cual puede ser invertido en la resolución de problemas fuera del área. Además, permite al usuario tener un acceso fácil a la información y en lo que concierne a la empresa aumentar la productividad.
Mediante la programación de estos mecanismos, se desarrollará una red neuronal artificial convolucional que, a través de la recolección de un conjunto de datos, se establece un entrenamiento que, conectada a un cámara, permite reconocer a detalle las características de la etiqueta que contiene la botella plástica.

1.4. Marco Teórico

El nacimiento del campo de la inteligencia artificial ha dado una mejora en todos los aspectos con los que se refiere a la interacción autónoma así puntualizan los autores (Rusell & Norvig, 2004), dando origen a realizar funciones complejas que requerían de mucho tiempo, en cuestión de minutos, en el área de la industria existe una mejora sin igual, en los procesos de envasado, empaquetado y distribución de productos, estas mejoras surgieron a mediados de los 80 donde cuatro grupos de investigadores generaron modificaciones sobre el algoritmo base de aprendizaje, en el que se aplicaba distintos problemas en las áreas informática y psicología. En el campo de la inteligencia artificial se ha venido desarrollando diferentes metodologías de investigación obteniendo así la creación de redes neuronales.

Las redes neuronales tienen una amplia línea de investigación, es decir, se puede aplicar en diferentes campos, una de las aplicaciones es para el estudio de determinados eventos o sucesos, en donde el tipo de aprendizaje se lo genera de manera supervisada, de esta manera, se forman diversas estructuras de redes con distintas funcionalidades. El uso de las redes neuronales permite detectar patrones dentro de un conjunto de datos relacionados entre sí, como se detalla en la en los siguientes estudios.

Un ejemplo de implementación, es la creación de una red capaz de saber a qué hora del día se utilizan con mayor frecuencia los artefactos, realizando por los autores (Cocconi, Yuan, Mulassano, & Ferreyra, 2019), se inició con el estudio de las actividades de los mismos, para ello antes de la programación del algoritmo, se generó un consenso en donde se analizó que tipo de red es la más apta. Teniendo como punto principal la detección de cualquier anomalía o variación existente en el sistema para su futura corrección.

También se puede realizar el análisis de imágenes mediante la aplicación de redes neuronales convolucionales, en donde pasan a través de filtros de modificación, para luego ser reconocida capa por capas, es decir, como se componen y que similitudes tienen cada uno de los pixeles convertidos en rangos de valores, para finalmente poder ser contrarrestado con un número, que dan probabilidades según los objetos dados como etiqueta. Existen investigaciones que se enfocan en este tema como se puntualiza a continuación.

Los autores (Martínez & Mingo, 2018) en su estudio sobre el campo del aprendizaje automático, manipulan simples líneas de código por medio librerías que son capaces de crear la estructura una red neuronal como es Tensorflow y junto a la máscara de Keras; la codificación realizada permitió reconocer las colas de las ballenas sin dejar de perder la calidad de funcionamiento. Inició con la obtención de imágenes en donde se enfocaron principalmente en el patrón que poseen sus colas.

Con respecto a la visión por ordenador que se ejecuta a la vez de una red neuronal se encuentra la investigación hecha por (Kanaan Izquierdo & Ventura Royo, 2016), en el que se trabaja la ejecución de reconocimiento de matrículas, siendo los sistemas expertos donde se implementó redes neuronales convencionales para el reconconocimiento de placas anteriormente cargadas en la base de datos, obtenido detalles y características de las mismas, se tiene como objetivo la ejecución de equipos electrónicos, cumpliendo con la función de automatizar procesos.

Una red neuronal desarrollada con la finalidad de reconocer cada señal de tránsito, fue realizada por (Távara Idrogo, 2019), en el que se llevó a cabo el uso de las redes neuronales

convolucionales, su finalidad era utilizar dicha red neuronal para realizarle distintas modificaciones las cuales daban paso a distintos modelos que tenían como finalidad ofrecer un resultado de mejora, se llegó a obtener un modelo con el porcentaje de predicción mayor al 99,5%.

Aquel modelo era capaz de reconocer cada una de las señales de tránsito con exactitud en donde se plantearía futuras implementaciones en vehículos automáticos que por medio de visión artificial puedan reconocer múltiples objetos a la vez.

Otra de sus múltiples aplicaciones es la visión por computador, por ejemplo, el análisis de imágenes en 3D, en gran parte se emplean en piezas que son de uso industrial. A pesar de ser un modelo destacado en los últimos años, su utilización se ha realizado de manera autónoma, siendo capaz de acoplarse con redes neuronales convolucionales con el propósito de extraer características esenciales, debido a que su visión es diferente a otro tipo de procesos, en el que su aplicación es de forma manual. Por lo tanto, se han generado distintas investigaciones en el que se verifica su efectividad.

Una investigación hecha por (Garrigues Carbó, 2018) señala que los sistemas que incorporan el uso de las redes neuronales convoluciones en las industrias, es para la inspección de la calidad del producto, es decir, que cumpla con los requerimientos y estándares solicitados por el cliente, mediante la implementación de cámaras, estas sirven de ayuda a la red neuronal en la obtención de imágenes, ubicándola a una distancia considerable.

Otro punto en el que se destaca el uso de redes neuronales es la mejora de los procesos, el autor (Medina Lescano, 2017) resalta la importancia de la automatización en máquinas, porque optimiza el tiempo y genera ganancias. Los artilugios tienen distintas funcionalidades, estos generan un colapso en el software para decisiones en las que operan sistemas expertos, por medio de redes neuronales capaces de realizar la toma de decisiones en tiempo real, sustituyendo la operación de supervisores, automatizando de manera instantánea procesos que por medio de la interacción humana requieren de mucho tiempo.

Redes Neuronales Artificiales

Las redes neuronales artificiales pretenden simular ciertas partes de las neuronas biológicas que tiene el cerebro humano, como lo es el manejo de impulsos eléctricos. Que al ser entrenadas estas aprenden hacer acciones, para las que están programadas. Cada una de las neuronas realiza un determinado proceso dentro de la red que está conformada por entradas con diferentes tipos de valores, estos pesos emiten un impulso en el umbral hasta llegar a la salida de la función (Antona Cortés, 2017).

Las funciones son iguales a los procesos matemáticos, se realiza el ajuste adecuado para cada neurona, teniendo independización una de la otra al momento de ejecutarse porque cada una funciona dependiendo de su peso tras su entrenamiento, por lo que en el procesamiento y manejo de datos se puede permitir la desconexión de una parte de estas sin perder el funcionamiento, haciendo que sea más óptima, por lo que aprenden exactamente los datos del enteramiento, dando resultados iguales, las cuales en este campo representan un problema, debido a que su finalidad es la de realizar sus propias decisiones independientes, evitando una igualación. (Martínez & Mingo, 2018).

Elementos de una red neuronal

La red neuronal se encuentra constituida por capas, nodos, pesos y bias. La finalidad de la red es que pueda aprender, de manera que el ajuste de los pesos se vuelve esencial y esto se realiza por medio de una función de activación que se encarga de evaluar múltiples valores de matrices en tiempo real, por lo que aquella parte siempre suele tomar mucho tiempo (Salas, 2005).

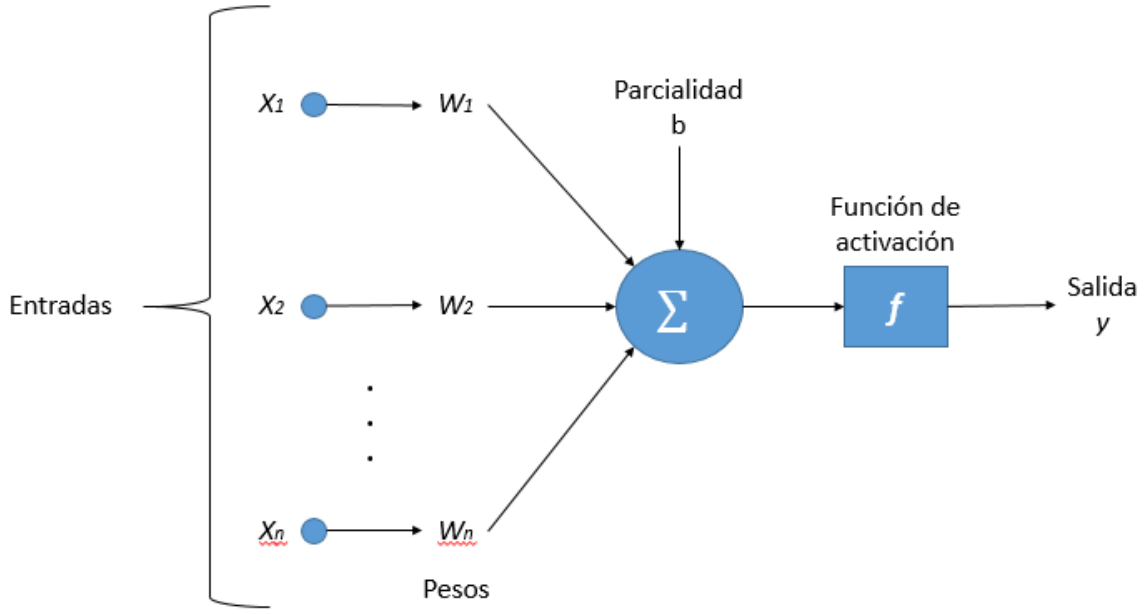


Figura 1:Componentes de una Red Neuronal Artificial (Maricalva, 2017)

Estas capas se encuentran interconectados entre sí y su traspaso de información es de manera constante, en su estructura principal se encuentran:

- **Capa de entrada:** Está compuesta por múltiples neuronas artificiales dependiendo del tipo sistema que se desee crear, cada una de estas representan a una característica importante del problema, la información que se le entrega a la red suele depender de

la estructura que se lleve a seguir, dando paso en la ejecución a interrelacionarse y reformulando su resultado para que sirva de alimentación en las siguientes neuronas.

- **Capas ocultas:** Son hileras de neuronas y cada una de esas obtiene el resultado de la capa anterior como datos de entrada, su funcionamiento consiste en operar sobre una función de activación, luego de esta terminan siendo ponderada sobre sus respectivos pesos que van ajustándose de una manera adecuada para poder facilitar su manejo de datos resultantes hacia la siguiente capa de la neurona.
- **Capa de salida:** Esta capa contienen un número de neuronas que serán dadas por el experto, estas representan a la salida que se desea dar, es el último filtro de clasificación en donde su función de activación suele estar configurada para que solo una de estas neuronas tenga una probabilidad superior al resto (Matich, 2002).

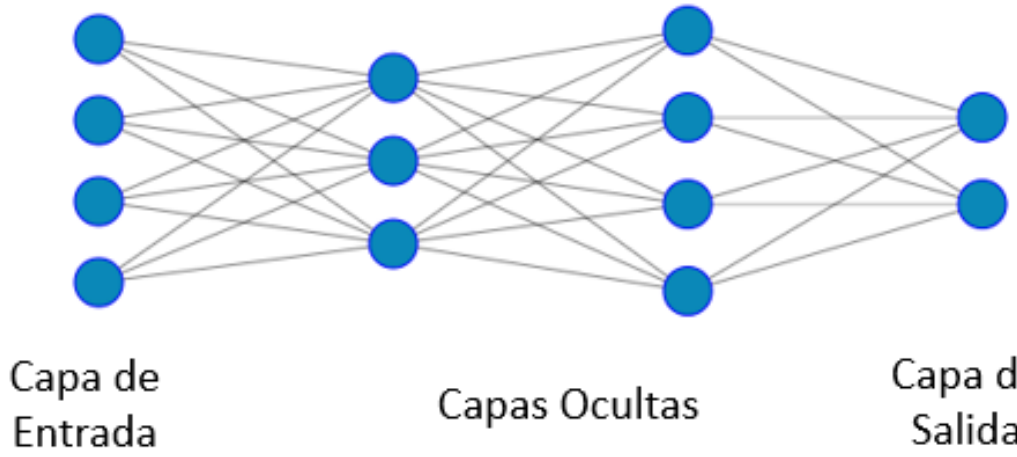


Figura 2: Arquitectura de una Red Neuronal Artificial

Red Neuronal Convolucional

A diferencia de una red neuronal artificial común que su manera de trabajar son solamente con datos numéricos, las redes convolucionales fueron desarrolladas para el aprendizaje a través de imágenes dado que cada pixel es representado por un valor que tiende de 1 a 255, la manera en que esta red trabaja es comprimiendo la imagen, compuesta en una matriz para reducirlas sin la necesidad de que pierdan sus puntos más importantes, para que al momento de pasar entre diversos filtros de convolución, la red logre aprender que cada rango de valores dados por la imagen representan el objeto que se desea buscar o descubrir simplificando el tiempo de entrenamiento en caso de que se quiera trabajar con una red

neuronal artificial sencilla (Pérez Carrasco, Serrano Gotarredona, Acha Piñero, Serrano Gotarredona, & Linares Barranco, 2011).

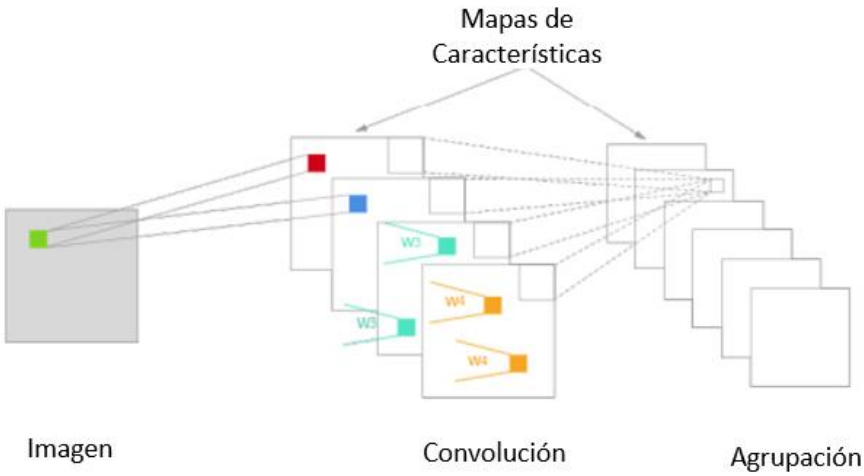


Figura 3: Configuración de una Red Neuronal Convolucional (de Benito Gorrón, 2018)

Las imágenes al ser de un formato visible para el ser humano, internamente no es más que una representación de mezclas de imágenes en formato RGB, donde tras la generación de un algoritmo realiza el desglose de la imagen en una matriz de colores, siendo cada una de estas una representación del color, a lo largo del entrenamiento, la red aprende a relacionar los colores con la imagen que se encuentre en ese momento. En estas neuronas su interconexión no es siempre estable ya que se sugiere como practica desconectar partes de aquellas para evitar un sobreajuste y hacer que la red tenga un óptimo procesamiento.

Estructura Básica de Red Neuronal Convolucional

- **Capa Convolucional:** Se dividen en dos tipos: la 1D se encarga de filtrar en bajo nivel una imagen, es decir, en blanco y negro. Mientras en que en la 2D maneja filtros a color, dando como resultado el aprendizaje de las neuronas para que estas relacionen diferentes filtros, para asimilar imágenes con variadas similitudes. Estas son de vital importancia cuando los datos a estudiar contienen diferentes escalas de colores.

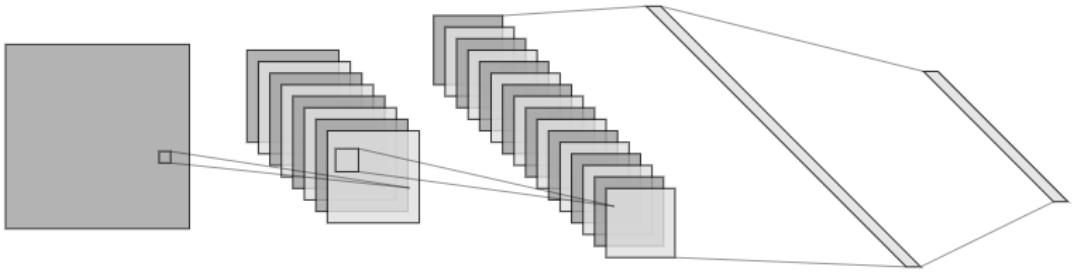


Figura 4: Funcionamiento de una Capa Convolucional

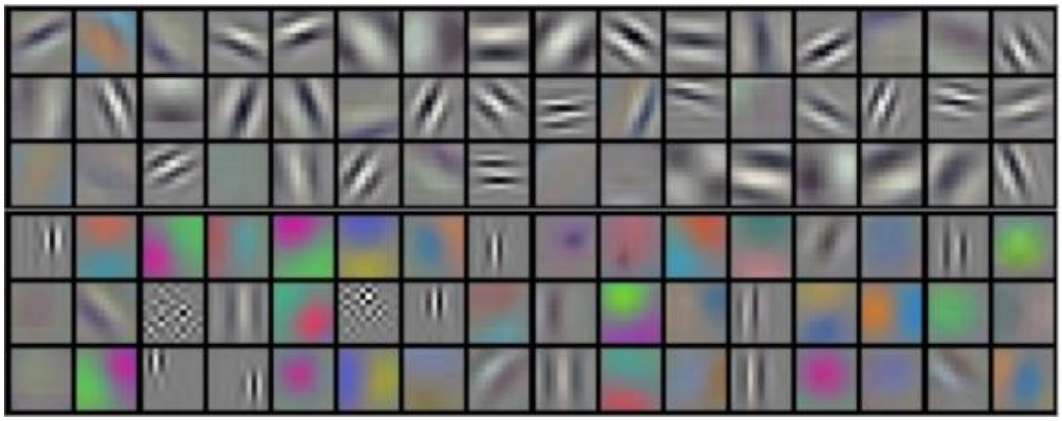


Figura 5: Filtros de la convolución (Pusiol, 2014)

- **Capa Pooling:** Se encarga de reducir la imagen, sin perder los detalles importantes, se clasifican en: max-pooling, la cual se encarga de buscar el valor máximo dentro de la matriz de pixeles, min-pooling, esta se basa en encontrar en número menor en una matriz de pixeles dando a resaltar los colores oscuros sobre los blancos y el average-pooling, el cual suma cada uno de los números dentro de la matriz y los divide para el total de campos que existen, esto hace que la imágenes a pesar reducir su tamaño no pierdan sus colores principales (Pusiol, 2014).

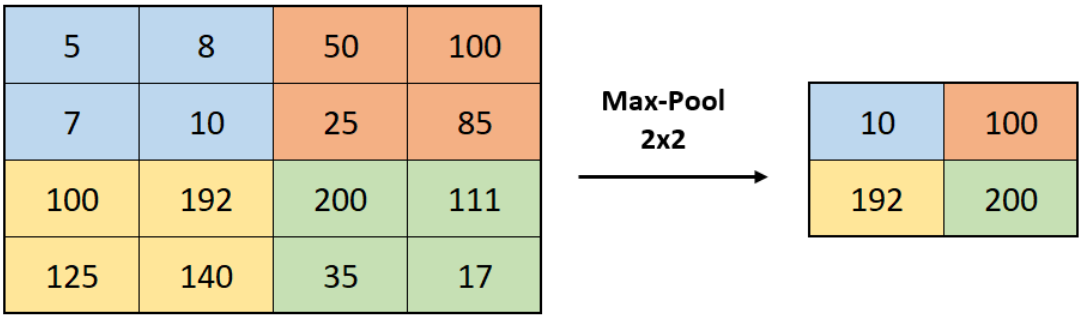


Figura 6: Función de la Capa Pooling

- **Capa Flatten:** Trabaja como capa intermedia la cual se encarga de realizar un traspaso de datos entre las convolucionales y las dense; cambiando los datos de una matriz a vector, permitiendo que se realicen otros tipos de funciones dependiendo de

la configuración que requiera el programador para la red neuronal (sigmoide, relu, softmax, etc).

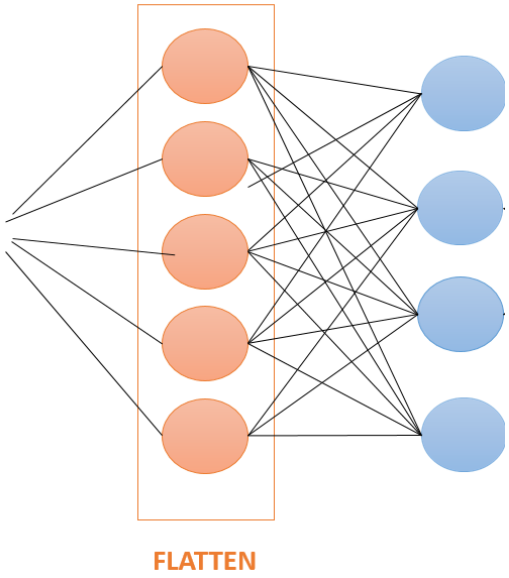
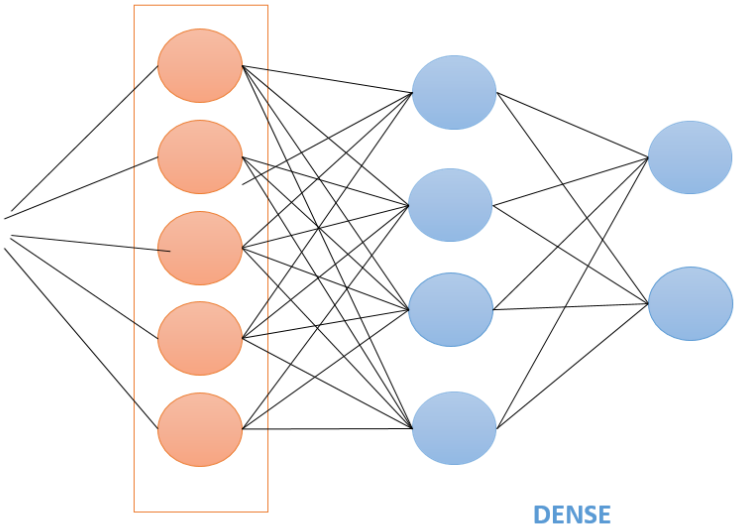


Figura 7: Estructura de la Capa Flatten

- **Capa Dense:** Se encuentran ubicadas al final de la red neuronal, todas las neuronas de dicha capa se encuentran conectadas entres si, junto con el peso que tiene cada neurona (Vilagran, 2018).



.

Figura 8: Arquitectura de la Capa Dense

Descenso del Gradiente

Es parte del modelo de aprendizaje de una red, utilizado comúnmente en el aprendizaje automático y que funciona como motor principal en gran parte de los sistemas de inteligencia artificial desarrollados actualmente, su funcionamiento está dado por continuas iteraciones, en el que él sistema tiende a ser estable, operan con la tasa de aprendizaje, teniendo como finalidad configurar los pesos de manera adecuada para que la clasificación de los objetos sea óptima (Garrigues Carbó, 2018).

Su algoritmo consta de la unión en todas las capas con sus respectivas neuronas, por lo que intentar ajustar los pesos de las neuronas de manera manual, requiere de complicadas técnicas matemáticas, tomaría mucho tiempo y esfuerzo, tornándose cansado y repetitivo. El estocástico es usando comúnmente en la retro propagación tornando de manera negativo al común descenso por gradiente (Durán Suárez, Del, Torres, & Suárez, 2017).

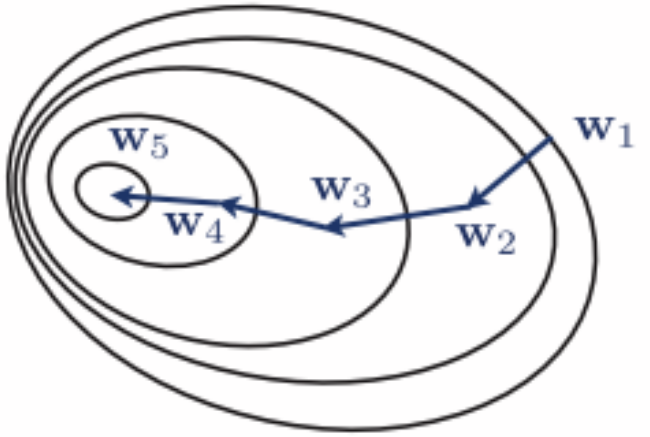


Figura 9: Iteraciones del Descenso de Gradiente (Mehryar, Rostamizadeh, & Talwalkar, 2012)

Función de Coste

Se encarga de detectar el error para cada una de las combinaciones existentes de los parámetros del algoritmo que se está creando. En Matemáticas tenemos las funciones como la convexa que cuenta con un punto mínimo global, la cóncava tiene un punto máximo, causando que no sea la más óptima y la no convexa es el resultado más común que poseen los algoritmos, en donde se encuentra múltiples puntos mínimos locales y un máximo global. La función de coste requiere aplicar la derivada para obtener la mayor pendiente en la posición actual, para luego usar el vector resultante, en sentido opuesto(Durán Suárez et al., 2017).

Función de Activación

Cumple con la funcionalidad de activar el comportamiento resultante de una neurona, estas se clasificarán dependiendo del resultado que se espere y son:

- **Sigmoide:** Dicha función otorga un resultado dado a través de una tangente hiperbólica y se encuentra formado por valores que van partidos en dos secciones, los cuales pueden ser dados dependiendo del rango que el sistema experto requiera y son de [0, 1] y [-1, 1]. Suelen ser utilizados cuando la red requiere de clasificar múltiples objetos que comparten similitudes haciendo que se ofrezcan probabilidades que dichos valores a ser estudiados.

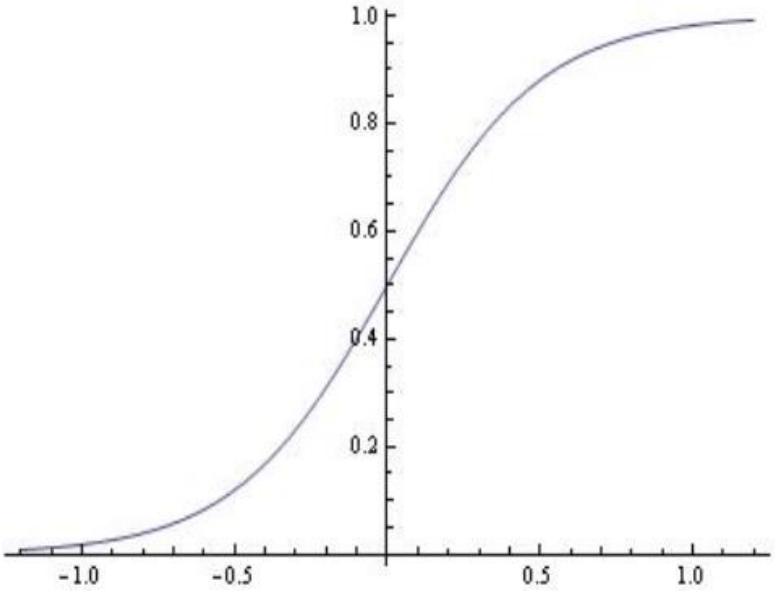


Figura 10: Grafica de la función Sigmoidal (Kyurkchiev & Markov, 2015)

- **Relu:** Se encuentra definida en un rango de valores abiertos que van de (0, X), donde la variable X es la representación de los datos de entrada a la neurona, supera a la función sigmoidal ya que su manera de acoplarse reduciendo el costo, se da de manera eficaz, pero se tiende a trabajar con una tasa de aprendizaje adecuada ya que, en caso de obtener una tasa un poco alta, la neurona puede ser capaz de quedar inutilizada en el proceso.

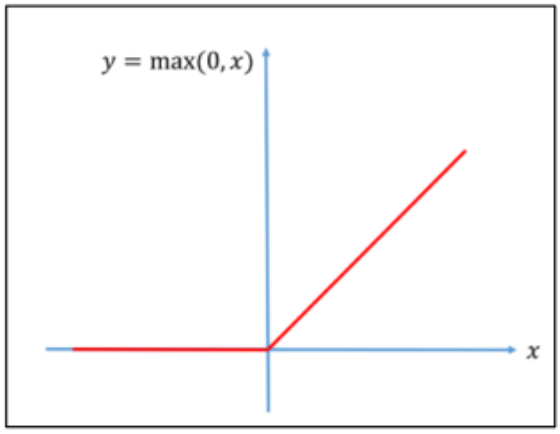


Figura 11: Grafica de la Función Relu (Arteaga, 2018)

- **Softmax:** Se maneja a través de puntuaciones, esta función se encuentra en la última capa de la red, donde se encarga de otorgar probabilidades a los objetivos resultantes, estos son dados a través de un vector que al finalizar el filtrado de los datos, el experto tiene que indagar en este vector e igualar las opciones de los resultados con los objetivos que se dieron de ingreso para su entrenamiento (Vilagran, 2018).

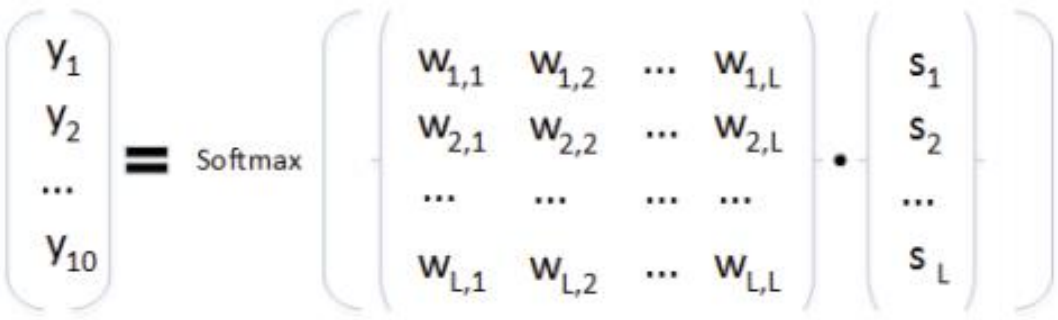


Figura 12: Resultado ofrecido por la función softmax (TORRICO RAMOS, 2017)

Sobreajuste

El sobreajuste o también conocido en inglés como overfitting es aquella que tiene exceso de flexibilidad en el modelo, es capaz de modelar incluso el ruido existente en los datos. Se especializa de forma correcta en su línea de regresión, en los procesos en el que ha sido entrenados anteriormente, es decir, llega a memorizar la solución del algoritmo. El sobreajuste no se preocupa en entender el conocimiento que subyace en dichos problemas. Este se puede observar cuando se trabaja con espacios en 2D y 3D (Maricalva, 2017).

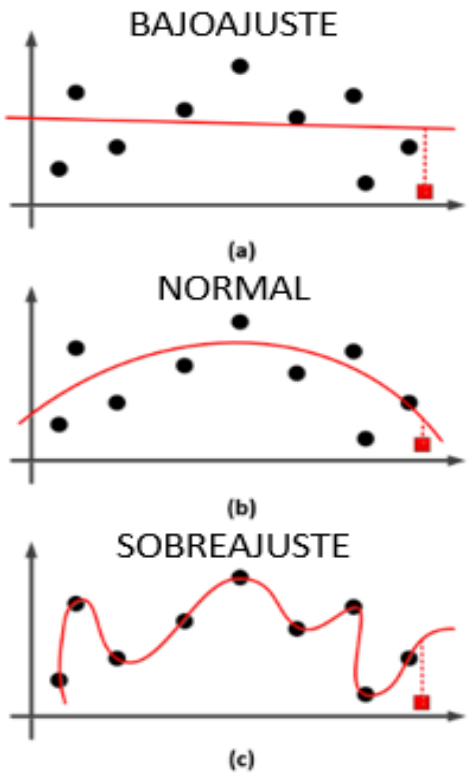


Figura 13: Graficas de Sobreajuste (Ghojogh & Crowley, 2019)

Otras de las maneras que se puede evitar el sobreajuste es usando un algoritmo ofrecido por tensorflow, el cual tiene que ver con la taza de aprendizaje que se usará en el proyecto. La función Adam es un optimizador que realiza luego de cada época un ajuste en la taza de aprendizaje, este ajuste dependerá de cual haya sido su respuesta en los pesos con respecto al último resultado dado, esta evita que la red realice por varias épocas el mismo resultado sin haber mejora dejando a la red aprenderse los resultados.

Herramientas

Python

Lenguaje de programación multiplataforma utilizado a nivel mundial por programadores seniors y juniors, con una interfaz intuitiva. Su manera de ser utilizado es capaz de ser multiparadigma, permitiendo codificar ya sea desde la orientación a objetos, imperativo y funcional. Manejando un estándar de codificación, logrando una adaptabilidad.

La relación del aplicativo de Python y las redes neuronales, es gracias al uso de diferentes librerías que son importadas, aquellas vienen creadas y modificadas por parte de las comunidades científicas, sin embargo, se puede realizar cambios en su estructura acoplándose a las necesidades del usuario, como lo es la librería matplotlib que permite al

programador realizar distintas graficas con los resultados entregados a través de variables (Kanaan Izquierdo & Ventura Royo, 2016).

Librerías de Python

Keras

Su finalidad es la creación de redes neuronales teniendo en la parte de la programación las librerías padres que son Tensorflow, Theano o CNTK funcionando en cada una de estas, en 2017 Tensorflow integro esta librería, obteniendo todo al alcance en una sola. Esto no significa que pueda trabajar con su librería padre, el cual funciona ingresando neurona por neurona, integrando funciones matemáticas que resulta tedioso para los especializados en la rama.

Con el pasar del tiempo ha venido mejorando su funcionamiento, tiene desde las más simple capa conocida como "Dense" que realiza las añadiduras de neuronas a una capa, permitiendo realizar la adaptación de funciones de activación con resultados esperados según las necesidades del programador, las capas convolucionales se expresan en 2D o 3D según el tipo de imagen que se presente, en el primero de los casos se encarga de reducir una imagen en escala de grises, para la segunda permite la reducción de las imágenes en escala RGB sin perder los datos de la original, permitiendo que la red neuronal tenga un procesamiento y entrenamiento más rápido de lo que se tendría si se realiza con la imagen en tamaño completo.(Antona Cortés, 2017)

Tensorflow

Librería desarrollada por Google, la cual consta de funciones que han ayudado al estudio de las redes neuronales para entrenar sistemas expertos. Optimiza la visualización y ejecución de grafos computacionales, trabaja con muchas capas, además permite programar funciones de activación y evalúa las predicciones que se obtiene, esta se usa para evaluar los errores existentes con el propósito de mejorar la red y minimizar el coste.

Las funcionalidades que incorpora esta librería, están enfocadas a múltiples herramientas desde la creación de redes, teniendo como finalidad los expertos en la rama, los cuales son capaces de manipular y realizar configuraciones externas, obteniendo como resultados avances en la tecnología, ya sea en el reconocimiento de imágenes o resoluciones a algoritmos matemáticos (Kanaan Izquierdo & Ventura Royo, 2016).

CAPÍTULO 2

2. METODOLOGÍA

2.1. Nivel de Investigación

2.1.1. Investigación descriptiva

En la investigación se aplicó la metodología de tipo descripta porque la profundidad de su estudio en las redes neuronales es de un nivel intermedio. Dentro de la clasificación de la investigación descriptiva se enfocó principalmente en estudios correlacionales debido a que su aplicación permite la relación entre variables manipuladas en el proyecto.

Se basará en las características de un conjunto de objetos para luego transformarlos en datos cuantitativos que sirvieron como fuente de alimentación para la red neuronal, codificando su estructura para un funcionamiento óptimo dentro de la mismas.

2.1.2. Investigación explicativa

Se aplicó la investigación explicativa para poder determinar las causas y los efectos de la problemática del proyecto para su correcto estudio. Debido a la falta de procesos que no son óptimos en la clasificación de botellas, no se realiza de forma eficiente la producción y categorización causando pérdidas en lo que se refiere la clasificación de los mismos.

2.2. Población

Para la población se realizó la selección de botellas plásticas, porque se encuentra bastante información sobre múltiples marcas existentes en el mercado actual, generando una fácil elección para el estudio.

2.3. Muestra

En la muestra se eligió un conjunto cerrado, obteniendo como objeto de estudio las marcas Coca Cola®, Sprite®, Fanta ®. Convirtiéndose en un grupo apropiado para el entrenamiento de la red neuronal.

Para la generación del modelo se tuvo en cuenta investigaciones similares realizadas anteriormente, que consistían en la adaptación de un sistema virtual a un enfoque interactivo, en donde el mundo real era participe como escenario en el que se desenvuelve la red neuronal.

El modelo que se llevará a cabo, necesita de las propiedades de los objetos a analizar, siendo estos las etiquetas de las botellas, programando en si una red neuronal capaz de satisfacer dichos requerimientos dados por la investigación, para esto se usarán distintos tipos de

librerías mencionadas con anterioridad en la investigación, ya que cada una nos permite generar distintos tipos de parámetros que como conclusión forman parte del todo.

Según la lista de características que tienen en las etiquetas, como lo son sus atributos el color de fondo, el tipo y color de la letra. Se desarrolló un dataset que corresponde a cada una de las marcas para que la red estudie y aprenda a identificar las etiquetas. Esta lista fue optimizada, debido a que un principio la creación de la red neuronal iba ser entrenada a partir de la forma de las etiquetas, siendo este conjunto de imágenes un total de 150 imágenes. Esto estimando un error en su funcionamiento causando confusión; porque los envases son similares una de otras. Por lo tanto, el dataset de imágenes se basa principalmente de las diferentes marcas de productos que tienen la compañía Coca Cola Company, y tras un estudio de los mismos se concluyó que los datos que se utilizarán eran netamente las etiquetas.

El sistema funciona bajo las características mencionadas anteriormente, por lo que usando el lenguaje de programación llamado Python, implementará famosas librerías en el campo de la inteligencia artificial, las cuales son Keras que se desarrolla bajo su código padre el cual es Tensorflow. Su interfaz gráfica estará desarrollada por Tkinter esta es una librería ágil y de fácil utilización con lo que se realizará para la entrada y salida de los datos. Las entradas son proporcionadas mediante dos formas, las cuales son: la obtención de una imagen a través del explorador de archivo (funcionalidad que implementa Tkinter) y la principal que es la obtención del entorno real por medio de una cámara, esta se mantuvo ejecutada desde la librería OpenCV, es la segunda parte de la columna vertebral del proyecto, enviando imágenes en tiempo real como parte de entrada a la red neuronal.

El diseño de la red neuronal es tomado por una investigación en donde el procesamiento de imágenes era abundante pero sus categorías u objetos era cortos, al fusionarse con las imágenes generadas, crean un único resultado el cual para la red cree conocer, esta a su vez contiene convoluciones encargadas en el reconocimiento de siluetas y aplicación de filtros. Al ser una foto común con dimensiones inmensas, estas deben tener un control de reducción, la cual es dada por una función dentro de Keras que reduce la imagen sin perder sus puntos fuertes, esto no es solo dado con motivos de memorias, sino que al reducir dichos pixeles liberan el sobre-cargamento en el proceso haciendo que dicha tarea requiera de varias horas más, comparada sino tuviera la función Pooling. Para evitar el sobreajuste en la red, una buena práctica es la desconexión de neuronas.

2.4. Sistema Propuesto

El siguiente diagrama se describe de qué forma el sistema realiza su función internamente, con las etiquetas de las botellas plásticas.

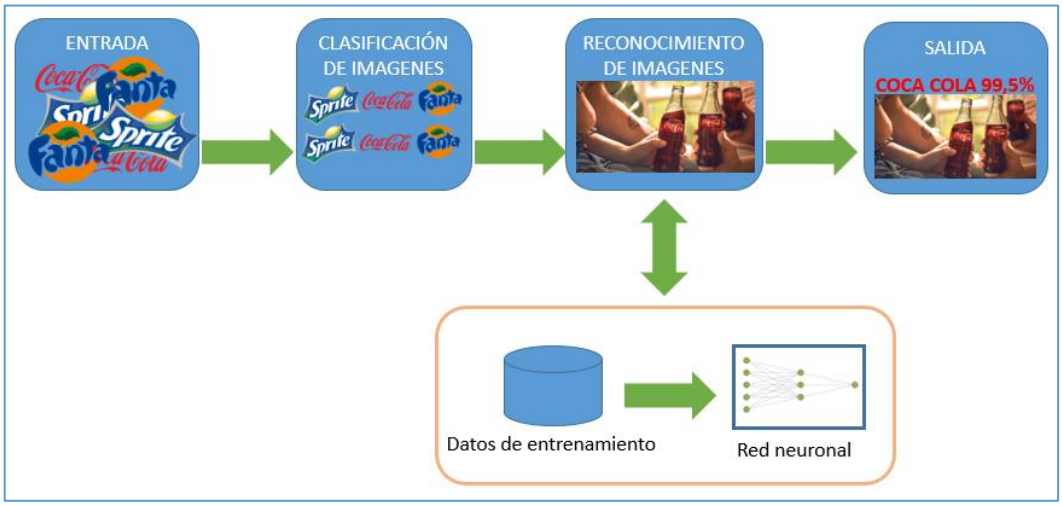


Figura 14: Estructura del Sistema

En la sección de entrada se trabajó con imágenes de las etiquetas de diferentes botellas plásticas, al momento de seleccionar, se tomó en cuenta las diferentes perspectivas y ángulos, en las que su tamaño puede variar, debido a que en las siguientes fases dicha imagen será tratada.

Dentro de la clasificación de imágenes se procede con la separación de la imagen a reconocer ya que la red permite que solo un logo sea encontrado a la vez con su predicción.

En la sección de reconocimiento de imágenes se realizó la detección de la etiqueta por medio de la red neuronal ya entrenada y posteriormente cargada los pesos, en donde la imagen ya clasificada redimensionada su tamaño, pasa por múltiples filtros y en la última capa de red neuronal; en donde la red procederá a arrojar un resultado con los pesos que ha aprendido.

La sección de salida está dada por la cámara que se encuentra en sincronización con el sistema, en donde en un recuadro arrojará en tiempo real que tipo de objeto es el que se encuentra ubicado para ser analizado.

2.5. Conjunto de datos de entrenamiento

Para el correcto funcionamiento de la red neuronal se requiere emplear una buena estructura y gran cantidad de datos para un mejor entrenamiento.

En la alimentación de la red neuronal se utilizó un dataset de 150 imágenes donde su formato está dado por un solo tipo que es JPG y su derivación JPEG, su almacenamiento se clasifica por secciones de carpetas en donde contienen sus respectivos nombres, que servirán a futuro como objeto de referencia para la identificación de la red. La selección de imágenes se

dividió en tres clases o tipos de etiquetas, en este caso se usó las más populares: Coca-Cola®, Fanta® y Sprite®.

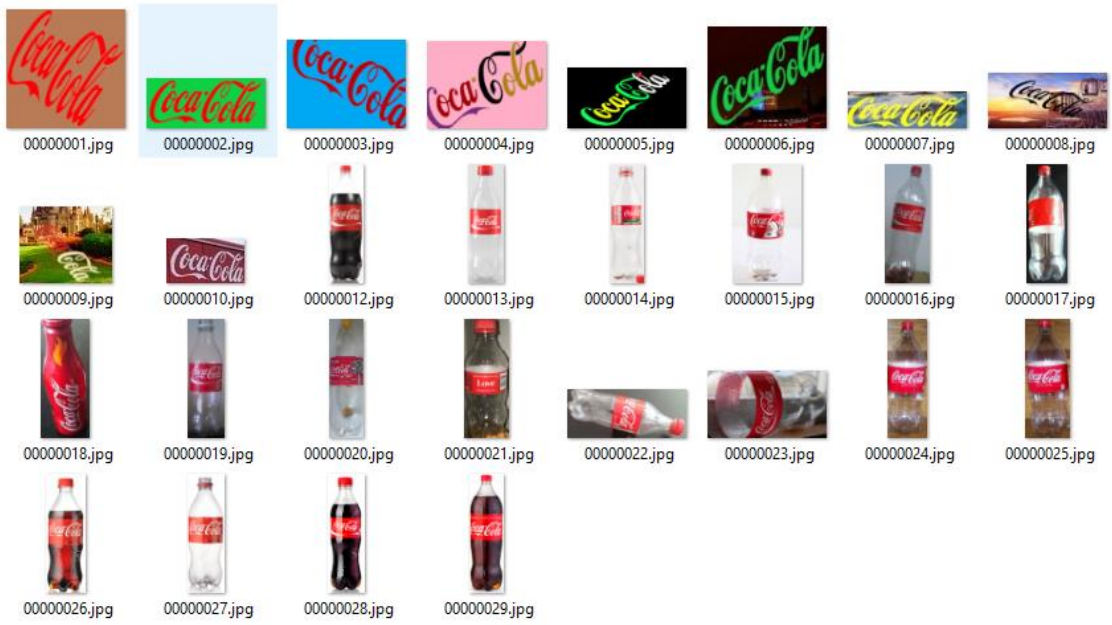


Figura 15: Conjunto de datos de entrenamiento de la Marca Coca Cola

Figura 16: Conjunto de datos de entrenamiento de la marca Fanta

Figura 17: Conjunto de datos de entrenamiento de la marca Sprite

2.6. Conjuntos de datos de validación

En la recolección de imágenes de botellas se considerará diferentes detalles, una marca de bebida contiene dentro de su inventario diversos tamaños de envases, por ende, la proporción de la etiqueta va a variar, dependiendo del modelo de la botella.

De las imágenes recolectadas un grupo se dividió para realizar la debida validación, porque la red neuronal convolucional se puede presenciar un sobreajuste, cuando las mismas imágenes de entrenamientos son usadas dentro del procesos de verificación.

Figura 18: Conjuntos de datos de validación de la marca Coca Cola

Figura 19: Conjuntos de datos de validación de la marca Fanta

Figura 20: Conjuntos de datos de validación de la marca Sprite

En la figura 19 se visualiza botellas de la marca Coca Cola, uno de los tres tipos de objetos a analizar, es de recalcar que las imágenes que son parte de la validación son diferentes a las de entrenamiento. Debido a que se trata de evitar la existencia de sobreajuste, para ello es necesario un 80% del dataset para el entrenamiento y el 20% para verificación.

2.7. Herramientas empleadas

Para el desarrollo y creación del aplicativo Python es una buena opción, que a través del IDE Pycharm proporciona lo requerido y facilita la programación al desarrollador, para una interfaz amigable la herramienta de Tkinter, como librerías útiles para el sistema a implementar las apropiadas son OpenCV, TensorFlow, Keras, Matplotlib, entre otras que se redactaran a continuación.

Python

Lenguaje de programación que contiene librerías creadas por diferentes investigadores, estos se relacionan con la inteligencia artificial y sus diversos conceptos, permitiendo al usuario trabajar de forma estructurada y rápida, convirtiéndose en la mejor opción al momento de programar una red neuronal.

Al importar diferentes complementos se puede generar pequeñas líneas de código que realicen funcionamientos específicos y de importancia, evitando de esta manera la redundancia y grandes códigos.

Librerías

En Python existen un gran número de librerías que son proporcionadas por desarrolladores e investigadores. Cada una de las librerías empleadas dentro de la estructura de la red neuronal son importantes debido a que proporcionan una serie de funcionalidades útiles, además sirven de base para otras librerías.

- La librería de Tensorflow se usó para las gráficas y ejecución de grafos computacionales además permitió programar las funciones de activación.
- Mientras que la librería de Keras se utilizó para la creación de la estructura de la red neuronal.
- El manejó de la librería de numpy sirvió para el cálculo numérico de matrices, suministrando todas las funcionalidades que estas tienen como realizar operaciones, invertirlas o hacer la transpuesta de ellas.
- Se empleó la librería scipy para ampliar el funcionamiento de numpy, utilizando herramientas científicas como el tratamiento de imágenes o datos, se relaciona de forma correcta con la librería anteriormente mencionada.
- Para la visualización de graficas se recurrió a la librería matplotlib.

2.8. Cámara

La cámara contiene una resolución de 2MP suficiente para adquirir una imagen y ser enviada a la red neuronal. No se utiliza cámara con conexión wifi, porque el envío de datos desde la cámara, que pase por el router o el pc y que sea interpretada tarda tanto, que la red podría colapsar por el envío a medias de los mismos.

Para una aplicación ya más industrial se requiere de cámaras con mayor resolución y que el envío de datos sea potente. Dando como resultado un mejor trabajo generando un alza mínima en productos clasificados.

Tabla 1: Características de la Cámara

Característica	***Especificación***
Resolución	*2 megapíxeles*
Integración	*USB*
Corriente	*5 voltios*
Funciones automáticas	*AES, AGC, AWB*
Sensor	*CMOS 1/3''*

2.9. Arquitectura de la Red Neuronal

Para la creación de la Red Neuronal, se empleó:

- En la capa de Entrada se usó 64 neuronas con configuración para que a partir de sus primeras interacciones comprenda pequeñas formas y filtros
- Se manejó 12 capas Ocultas en las que se mezclaba 3 tipos de redes, siendo la primera las convolucionales explicada con anterioridad para que la red pueda mejorar con una mayor búsqueda de filtros y siluetas. Estas a su vez interactuarán con el entorno dando como resultados pesos que se irán graduando con el pasar de cada época. La segunda capa de MaxPooling se encarga de reducir el tamaño de la imagen matemáticamente sin que se pierda su calidad, haciendo que la red interactúe con la imagen, esto permite que para cualquier tipo de grafico o ilustración sea procesada. Esta técnica fue implementada en tensorflow sobre sus últimas actualizaciones porque en proyectos anteriores a dicha versión, se usaban librerías como OpenCV para su modificación. La última red es Flatten, es la conexión de las redes convolucionales que se manejan en 3 dimensiones a una sola para seguir con el entrenamiento debido.
- La capa de Salida es la encargada de generar la clasificación para la construcción de resultado que contiene un total de 4, cada neurona representa a la salida del objetivo que en este caso serán las etiquetas: Coca Cola, Sprite, Fanta y como una etiqueta que se manejara por defecto será el fondo donde se aplicó el escenario, es de categoría Dense, la cual es capaz de aplicarse métodos de activación dedicadas a múltiples trabajos, esta da una síntesis de lo que se desea buscar permitiendo que los resultados dados por la red puedan ser utilizados y aplicarse metodologías de desarrollo dependiendo del propósito del trabajador.

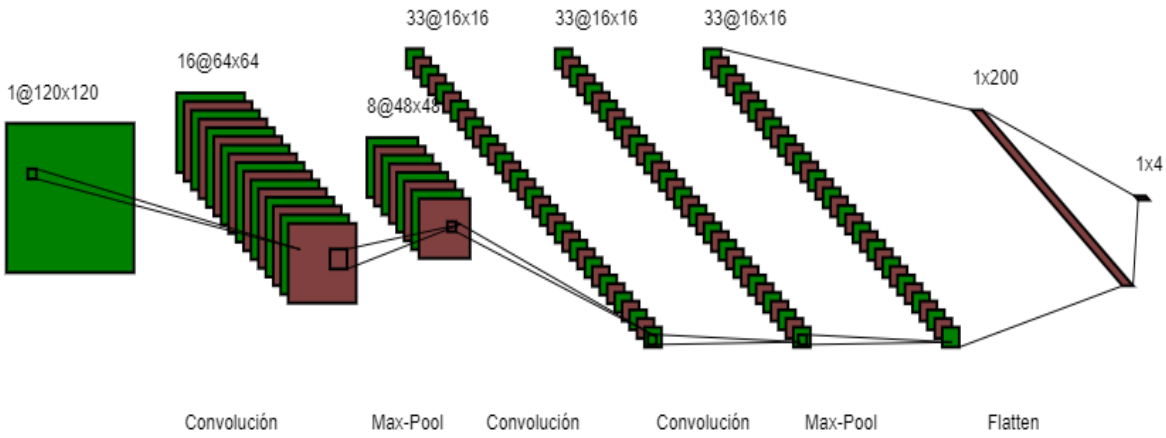


Figura 21: Estructura de las capas de la red neuronal del aplicativo

Se requirió de las funciones de activación para que las neuronas puedan accionar, estas a su vez permiten guardar su peso que creen la mejor optimización a la hora de ser puesta a prueba. En la red se aplicó:

- *Sigmoide:* Se usó como función para realizar el cambio de la imagen las cuales van en colores a simples datos generados en matrices, que por siguiente sirvieron como fuente de alimentación en el ámbito de entrenamiento a la red.
- *Relu:* Fue aplicado en múltiples estilos de redes neuronales, ya que esto permite que la red pueda arrojar resultados con valores altos. esta función otorga la libertad de arrojar resultados altos para su configuración en pesos.
- *Softmax:* Aplicado en la red neuronal, funciona como clasificador cuyo resultado deja al programador la oportunidad de implementar sobre su interfaz gráfica (lo que se desea esperar). Usualmente es igualado con los nombres de los objetos escritos binariamente para su selección apropiada.

CAPÍTULO 3

3. RESULTADOS (ANÁLISIS O PROPUESTA)

3.1. Tema:

Desarrollo de una Red Neuronal Convolucional para la Clasificación De Botellas Plásticas en una Empresa Envasadora de Bebidas Gaseosas de la Ciudad De Milagro.

3.2. Descripción de la Propuesta

Desarrollo de una herramienta que contiene una red neuronal convolucional, que permite clasificar las botellas plásticas, basando en los atributos que contiene la etiqueta del envase por medio de un dispositivo de transmisión de video, con la finalidad de reducir tiempo y mejorar rendimiento de los procesos de categorización de los productos.

3.3. Estructura del sistema

En el siguiente diagrama se visualiza el funcionamiento con respecto a la interacción del software y hardware, iniciando por el análisis del objeto mediante el uso de una cámara usb que obtiene el dato que se enviará al sistema para su futura predicción. Una vez la red haya predicho el objeto que se encuentra frente a la cámara, ofrecerá un resultado para el usuario que consistió del nombre de la etiqueta, junto al porcentaje de aceptación.

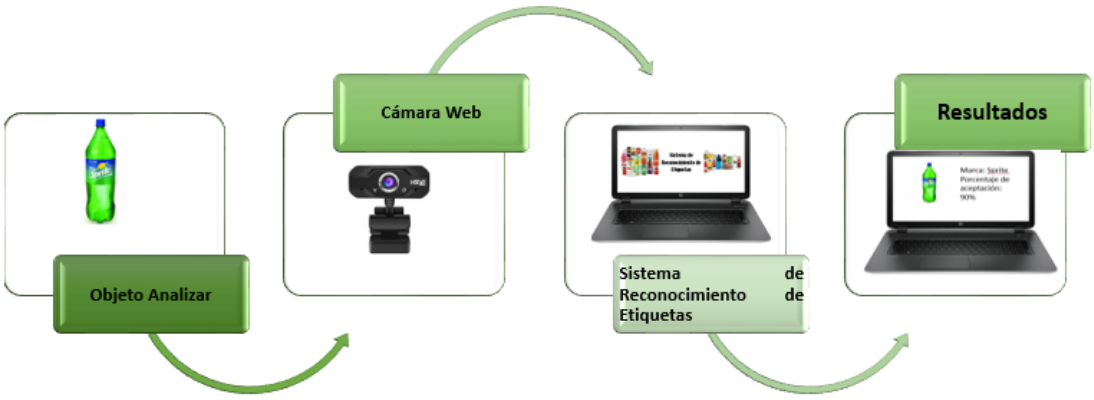


Figura 22: Diagrama del funcionamiento del sistema

El diagrama interno de como realiza cada una de las interfaces, se creó un diagrama de flujo en que consista la demostración de cada una de las pantallas con respecto a la interacción que realiza el usuario y que lógica sigue con cada una de la reacción.

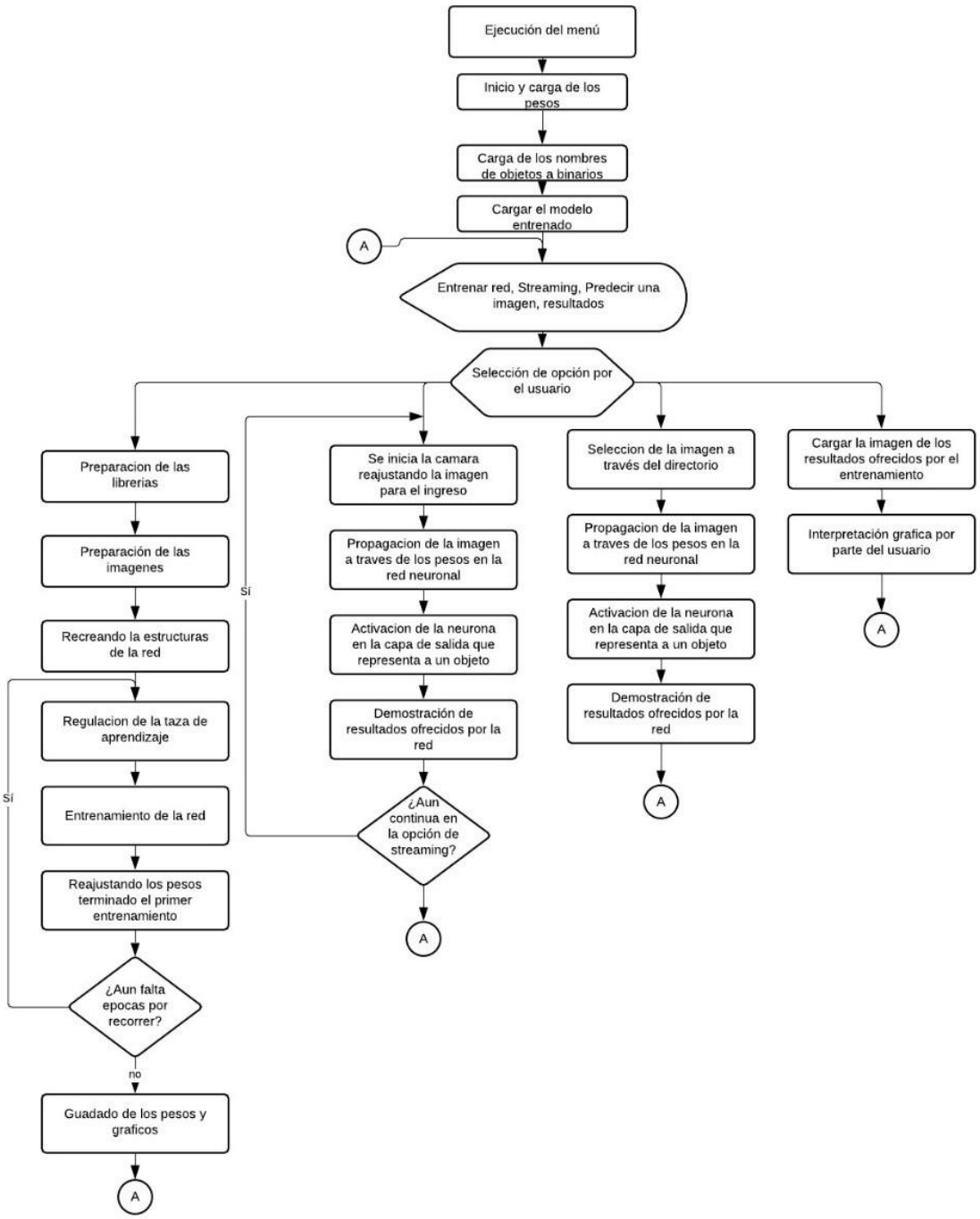


Figura 23: Diagrama de flujo correspondiente al funcionamiento

3.4. Evaluación del entrenamiento de la red neuronal artificial

Para conocer cuáles son las capacidades de la red neuronal, la red tuvo un direccionamiento al dataset que consistió de un total en 80% para el entrenamiento y un 20% para validación,

esto fue dado debido a que en la mayoría de investigaciones anteriores enfocadas al análisis de imágenes de la red neuronal se usa este tipo de particiones en los datos.

Para la generación de la estructura de la red neuronal, se realizaron practicas con redes empleadas en anteriores investigaciones realizando cambios sobre las mismas, séase la adición o sustracciones de neuronas o capas ofreciendo distintos valores en su error permitiendo que se pueda establecer la red fija con la cual se está trabajando mientras que a continuación se mostrara cada intento en búsqueda de una red óptima.

Tabla 2: Tabla de entrenamiento con distintas estructuras

Estructuras de la red (N° de neuronas por capa)	**Numero de épocas de entrenamiento**	**Error**	**Observación**
64-64-128-128-1024-4	100	0.12	Descartada
64-32-128-1	10	0.5	Optimizar
3-64-64-64-128-128-128-256-256-256-512-512-512-512-512-512-4096-100	10	0.4	Optimizar
30-64-64-128-128-512-512-1024-1024-500-500-100	10	3.2	Descartada
3072-1000-500-100	10	3	Descartada
3-64-125-300-250-1024-4	10	4	Descartada

Luego de obtener la red a través de varias pruebas se procedió con el entrenamiento, en el que consistía en experimentar y deducir cual es el mejor número de épocas para un correcto funcionamiento, para la primera prueba se realiza con un total de 50 épocas, su perdida era inestable haciendo que en ciertas épocas sus valores sean altos o bajos, su error se encontraba cerca de 1 como se visualiza en la Figura 24**,** haciendo que su resultado al validar con el error sea de valores casi que aleatorios ofreciendo una respuesta menor al esperado por lo que se determinó que no era la más óptima.

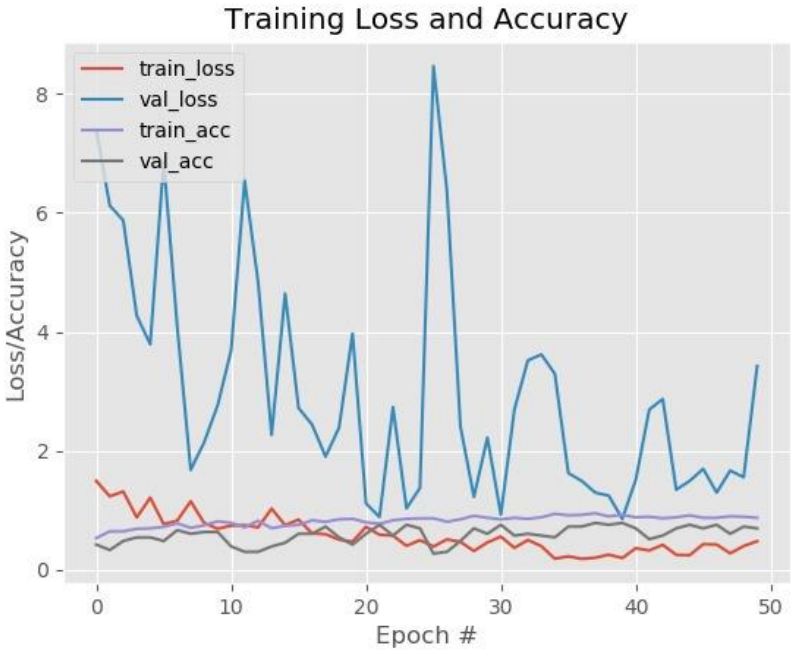


Figura 24: Entrenamiento de red con 50 épocas.

Cuando se intentó aplicar un máximo de iteraciones como lo es 200 épocas como se puede observar en la Figura 25, los valores que la red ofrecía eran más aleatorios en los puntos que se deberían conocer como estables, su error estuvo rondando casi que 0.5 pero al aumentar a 200 su iteración brinca cerca de los 160 lo cual solo deja que para el siguiente entrenamiento se considere un numero de 100 para obtener resultados estables.

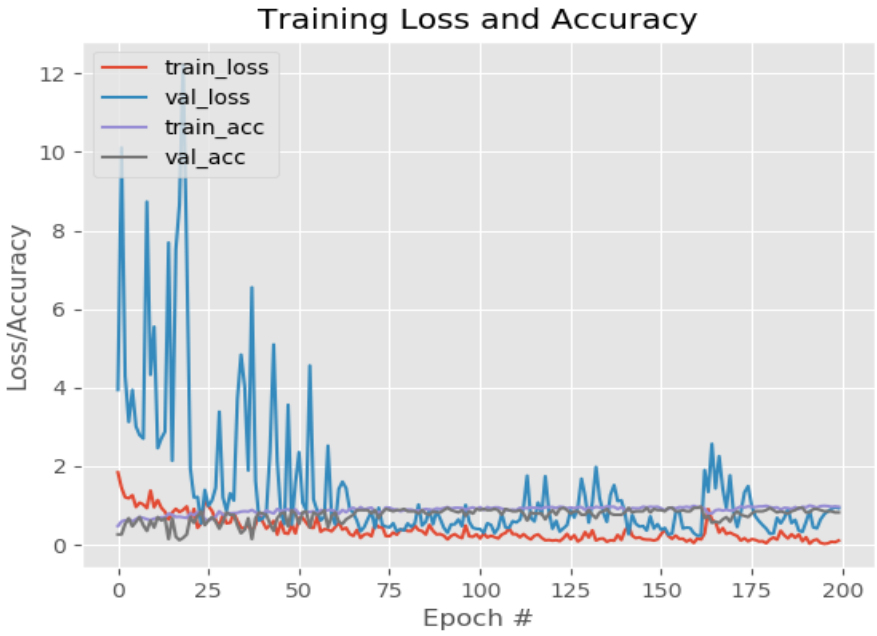


Figura 25: Entrenamiento de la red con 200 épocas.

Para la mejora continua, se le otorgo un total de 100 iteraciones, porque al proporcionarle una gran cantidad de épocas es capaz de que entre al sobreajuste y se pierda toda la estructura de la red y obtener resultados metodológicamente erróneos.

Como todo entrenamiento requiere de una tasa de aprendizaje, esta es proporcionada por tensorflow la cual es Adam, este es un algoritmo de aprendizaje interaccional. Luego de cada época la tasa tiende a cambiar, si necesita decrecer o crecer dependiendo de los resultados actuales que ofrece la época, para hacerse iterativamente óptima.

Se inició con un conjunto de imágenes pequeñas, alrededor de 150 en las que se puede observar la etiqueta de distintas marcas de botellas, causando que el entrenamiento de la red sea ineficiente y que el programa no funcione de forma correcta. Por lo tanto, se amplió el dataset implementando nuevos enfoques y ángulos, los cuales tras su aprendizaje ofreció un óptimo resultado, debido a que Python a través de sus funciones genera una serie de configuraciones internas.

Al momento de estructurar la red se probó que la función de activación sigmoide arrojó resultados en cada una de las capas neuronales con rango de valores entre -1 y 1, abriendo paso a las neuronas a dar una respuesta que no es óptima concorde a la última capa, por lo tanto, no forma parte de la arquitectura final de la red neuronal.

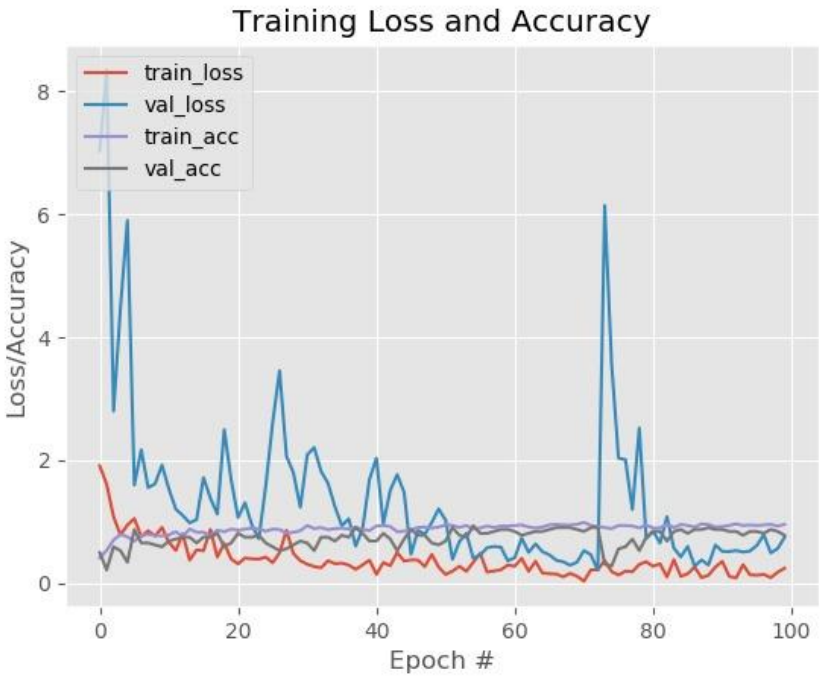


Figura 26: Resultados de las perdidas con respecto a la época.

Como resultado de la red se pretende minimizar el error con respecto a la época, por eso a mayor número de épocas, menor es el error ya que la red aprende todo lo relacionada a la misma, en las primeras 60 épocas la red tiende a ofrecer resultados casi que aleatorios, siendo en ciertas épocas un salto que se puede apreciar, pero a partir de allí el error el cual varía entre 0 y 2, corresponde a la línea azul la cual es la validación. En lo que corresponde al entrenamiento (Fig 26, grafica de color rojo) su línea grafica está casi que, rozando cero, ya que es lo más estable que puede haber en resultados. Con respecto a las validaciones y entrenamiento son estables ya que es la cantidad de tiempo que el error toma en establecer un mínimo cerca del global, estos valores no son importantes para la deducción del sistema que se está creando.

Tras probar distintas redes neuronales con múltiples funciones de activación, se conoce el resultado que ofrece cada uno por lo que se llegó a constatar que las más optimas que realizan el correcto entrenamiento son las siguientes:

Tabla 3: Funciones de Activación Aplicadas

Funciones de activación	Ecuación
ReLU	$f(x) = \ln(1 + e^x)$
Softmax	$\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}\ para\ j = 1, \dots, K.$

3.5. Resolución de la imagen y distancia de la cámara

Para el ingreso de la imagen se propuso como es una pequeña red neuronal comparada a una que ha tenido un gran número de entrenamiento con grandes dataset y su entrenamiento correspondiente a su conjunto de datos que no es demasiado, su resolución de imagen para el tratamiento y distribución es de pixeles es de 29x29, lo cual es muy poca la resolución de la imagen, siendo que esta dependerá de la cámara, este periférico es de 720x420 pixeles pero al ser otorgada antes de la red por medio de las librerías que tiene Python es tratada para que se reduzca y pueda ser de dimensión correcta para la red.

Una distancia considerable para la visualización a detalle de la etiqueta es un máximo de 7 centímetros para que la red encuadre netamente la marca, si se aleja más el objeto la red no podrá detectar el objeto que se encuentra frente de la cámara y ofrecería un resultado erróneo y al encontrarse más cerca de esta distancia se perdería el enfoque del objeto.

Con respecto a la distancia que debe existir entre la pared y la botella es un total de 18 centímetros, esta puede variar ya que se puede extender por todo el ancho de la plataforma, siempre y cuando la distancia entre la cámara y la etiqueta.

3.6. Manual del sistema

Requerimientos técnicos para el uso:

- Python 3.6
- Procesador Intel Core i3 (Para el entrenamiento)
- Sistema Operativo Windows 10

A continuación, se muestra el menú principal del sistema:

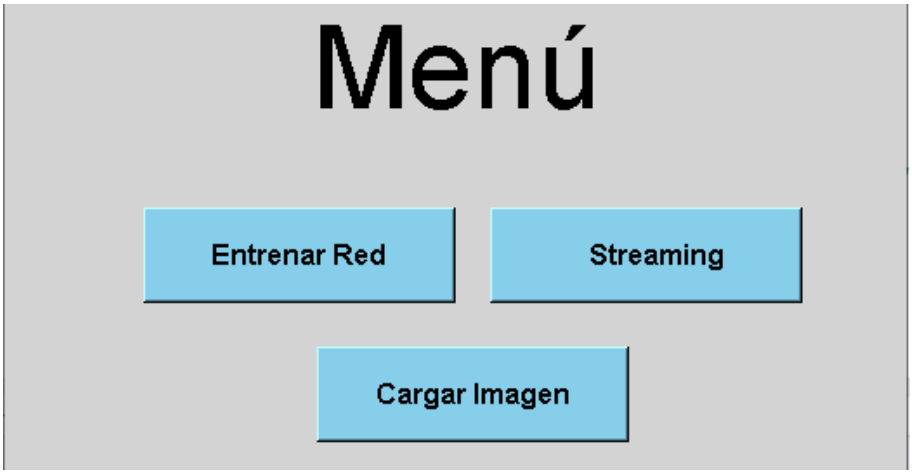


Figura 27: Módulos del Sistema

En la parte central de la pantalla se muestran 3 opciones que describen el funcionamiento del sistema. Para iniciar con el entrenamiento de la red neuronal convolucional se selecciona en **Entrenar Red.**

Una vez presionada la opción **Entrenar Red** el sistema procede a administrar al usuario la interfaz de entrenamiento.

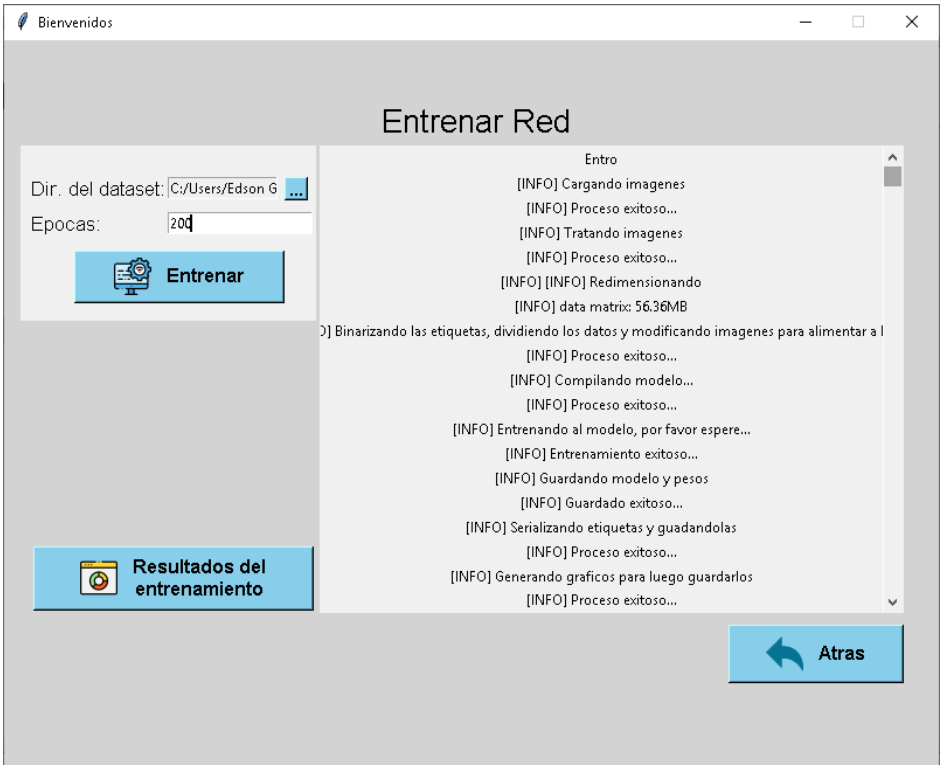


Figura 28: Pantalla de entrenamiento de la red neuronal.

En el apartado **Dir. Del dataset** se debe selecciona a través del Explorador de Archivos, el nombre de la carpeta donde se ubican las imágenes que serán tomadas en cuanta para el proceso. Seguido a esto se en la parte de **Épocas** se colocará una cantidad con las que se desea trabajar, pero podrá ingresar un máximo de 300 épocas, por lo tanto, el sistema tiene validada dicha opción.

El botón con el icono  procede con el arranque para el entrenamiento de la red neuronal, pasando por las validaciones en caso de no tener alguno de los dos datos ingresados. Ya ingresado y presionado el botón el sistema procede a mostrar en el recuadro blanco parte del entrenamiento realizado por la red.

El icono  retorna al usuario a la interfaz del menú principal, cancelando todas las acciones realizadas por el usuario dentro de la opción.

En la opción **Resultados del entrenamiento** dada por el grafico  redirige al usuario a una interfaz donde se mostrará los gráficos estadísticos dados por el entrenamiento.

Figura 29: Pantalla de resultados estadísticos

Dentro de la interfaz de **Gráficos Estadísticos se** mostrará al usuario los datos cuando presione el icono estos son los datos proporcionados por la red una vez se produzca el entrenamiento, están en formato de imagen, son dados para que el usuario pueda ver el comportamiento de las perdidas y otros resultados dados en cada época previamente asignada por el usuario.

Al presionar el icono permite retornar al usuario a la interfaz del menú principal que tiene el sistema.

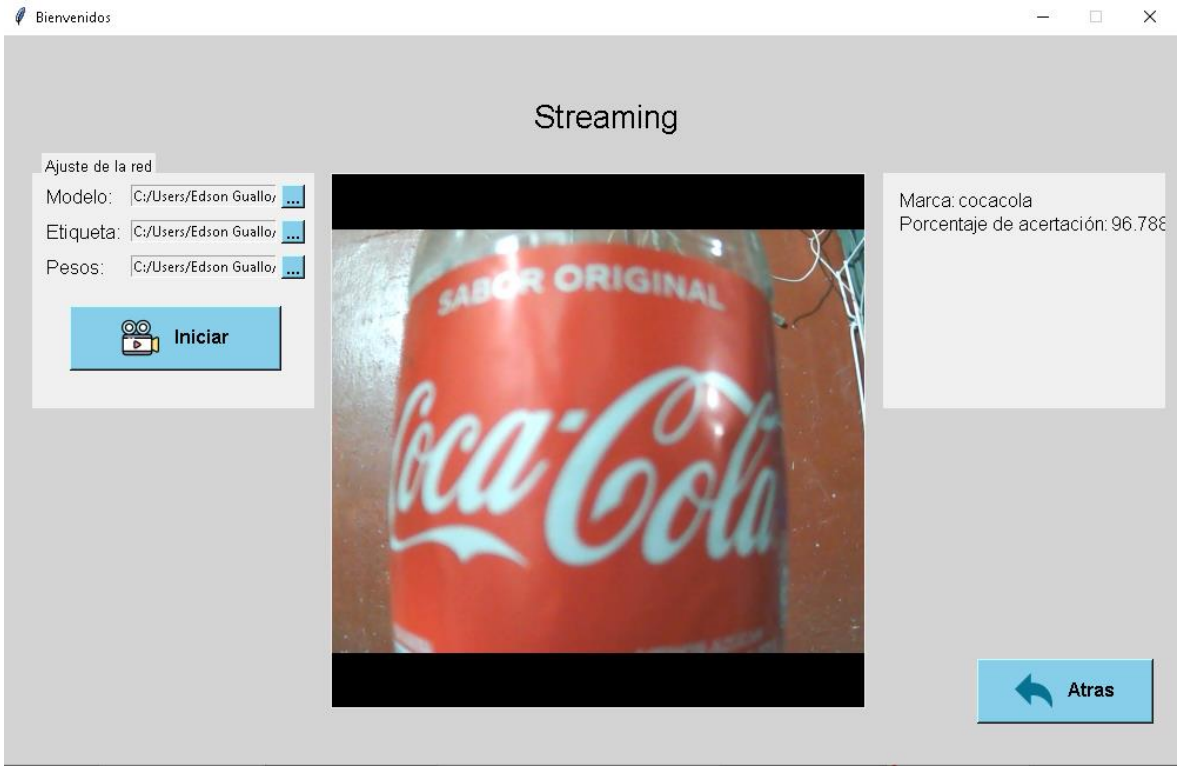


Figura 30: Interfaz de la captura de imagen en tiempo real

En el módulo de **Streaming** se debe llenar los campos de modelo, etiqueta y pesos seleccionado los archivos correspondientes a cada opción, para posteriormente encender la conexión de la cámara con el sistema, es importante recalcar que es obligatorio llenar todos los datos, porque esta información es requerida para el funcionamiento de Streaming.

Al presionar  **Iniciar** se establecerá el enlace entre el aplicativo y la cámara, se ubica el objeto analizar al frente de la cámara y automáticamente el sistema arrojara un mensaje de que tipo de marca es la botella y que porcentaje de aceptación tiene.

Esta pantalla también cuenta con el botón de **Atrás**  para volver al menú principal y seguir interactuando con el siguiente modulo.

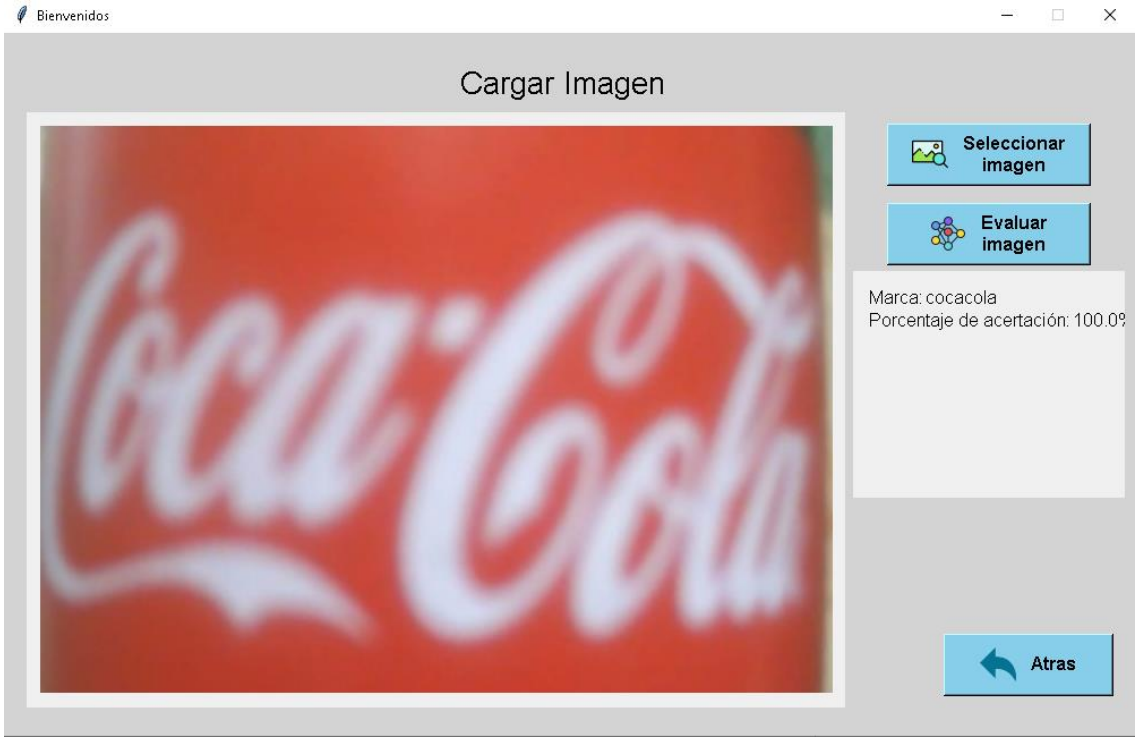


Figura 31: Pantalla de carga de imagen.

Cargar Imagen permite que la red neuronal convolucional examine una determinada imagen seleccionada en el archivador de su computador y presentar si es o no una botella lo que se cargó en el sistema.

El botón de **Seleccionar Imagen**  direcciona al explorador archivos que tiene el sistema operativo, para seleccionar una imagen, el sistema no ejecutara la evaluación si detecta que no existe una imagen que analizar.

A presiona el botón de **Evaluar Imagen**  **se** iniciar el proceso de análisis de la imagen y como resultado se presenta un mensaje con la marca y porcentaje de aceptación.

Para volver al menú principal se selecciona el botón de **Atrás**  donde redireccionara al usuario hacia dicha interfaz.

3.7. Pruebas realizadas en el prototipo

Los resultados son parte de la interacción del sistema con el prototipo contando tanto para la carga de imágenes y el streaming, siendo el más fuerte el ultimo mencionado ya que al ser

la ejecución en tiempo real este tiende a ofrecer distintos resultados a parte del objeto que este capturando la cámara, los detalles se muestran a continuación.

Tabla 4: Resultados de reconocimiento

Marca	**Cantidad de botellas utilizadas**	**Porcentaje de observación**	**Observación**
Coca Cola	7	97%	De los ejemplares a tomar, 4 resultaron con una buena probabilidad de aceptación, 3 con una menor y 1 no fue reconocida por el sistema. El color que emplea la marca es fuerte así que es normal que sea reconocida con facilidad.
Sprite	5	85%	Resultaron que 3 fueron acertadas con una probabilidad estable y 2 resultaron con una probabilidad menor esto se debe a que su color que enmarca es verde este al ser uno de la escala representante en RGB es fácil su reconocimiento
Fanta	6	72%	Del grupo analizado 5 fueron acertadas con probabilidad baja y 1 no fue reconocida esto puede ser a la mezcla de dos colores que forman su etiqueta siendo el naranjal un color que depende de la iluminación para tener un fuerte o débil color

3.8. Analisis re resultados

En la interfaz principal, permite al usuario interactuar con múltiples opciones que dispone la red neuronal artificial para su ejecución.

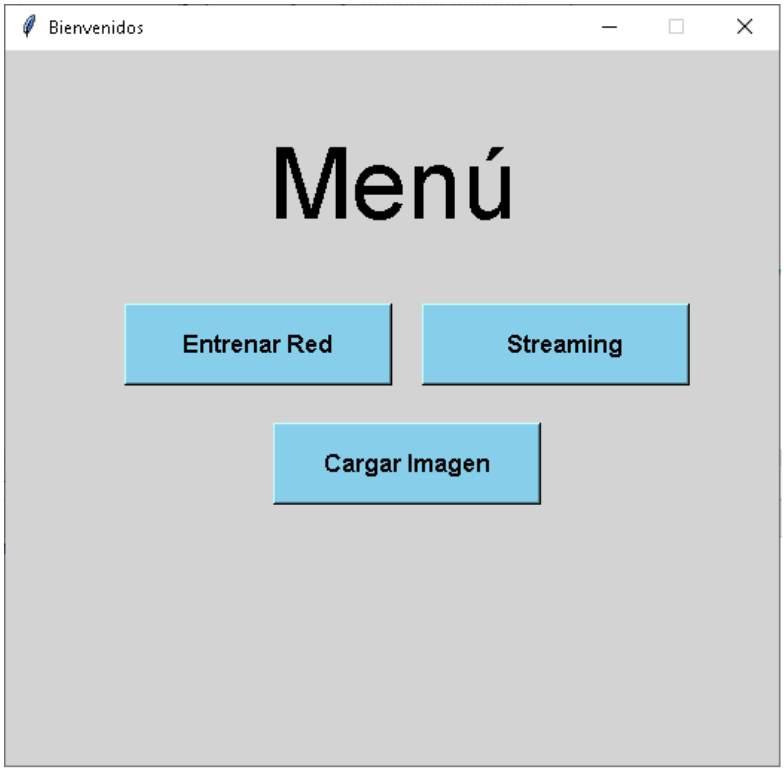


Figura 32: Menú principal de la interfaz

La pantalla de Entrenamiento de la Red permite la selección y se especifica cual será la carpeta que alojará las imágenes para el entrenamiento, para la especificación de épocas para las sucesiones del entrenamiento.

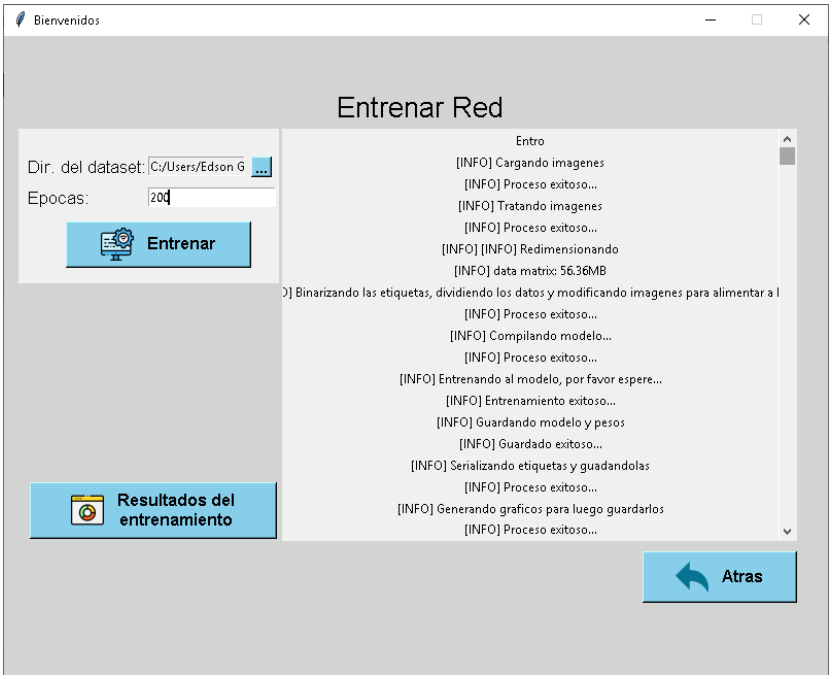


Figura 33: Interfaz del Entrenar Red con sus resultados

En la sección de gráficos estadísticos es en donde se muestran los resultados de una red entrenada para las observaciones a expertos.

Figura 34: Generación de Gráficos Estadísticos

Al momento de Cargar la Imagen se procede con la selección de las imágenes en donde se continua con el guardado de la dirección y se muestra la imagen elegida en el apartado de la pantalla, después con la evaluación se envía la imagen a la red ya entrenada para que prediga cual es la marca mostrada en la botella, como se puede observar en la figura 30 el sistema envía un mensaje de que la marca es CocaCola y que el porcentaje de aceptación es de un 99.6% de la imagen analizada.

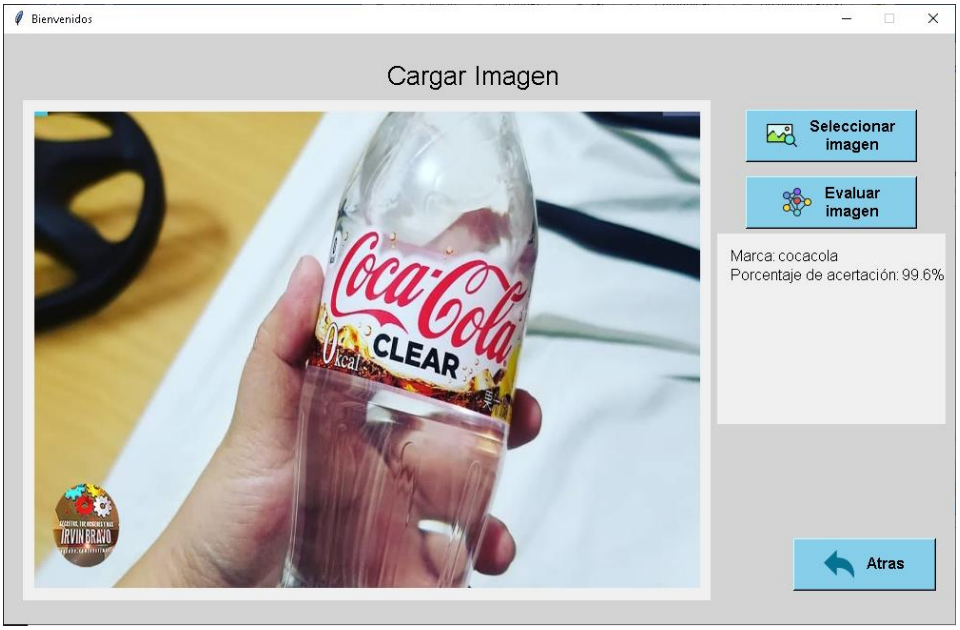


Figura 35: Ejemplo del análisis de una imagen en el aplicativo

El sistema está diseñado para también detectar cuando la imagen a ser estudiada no se trata de una botella, dando como mensaje de que no existe las botellas con un porcentaje igual o menor del 80%, es decir, que la red ofrece resultados hacia una marca, pero son incorrectos, por lo que se aplicó un filtro con probabilidad de aceptación igual o superior al 97%.

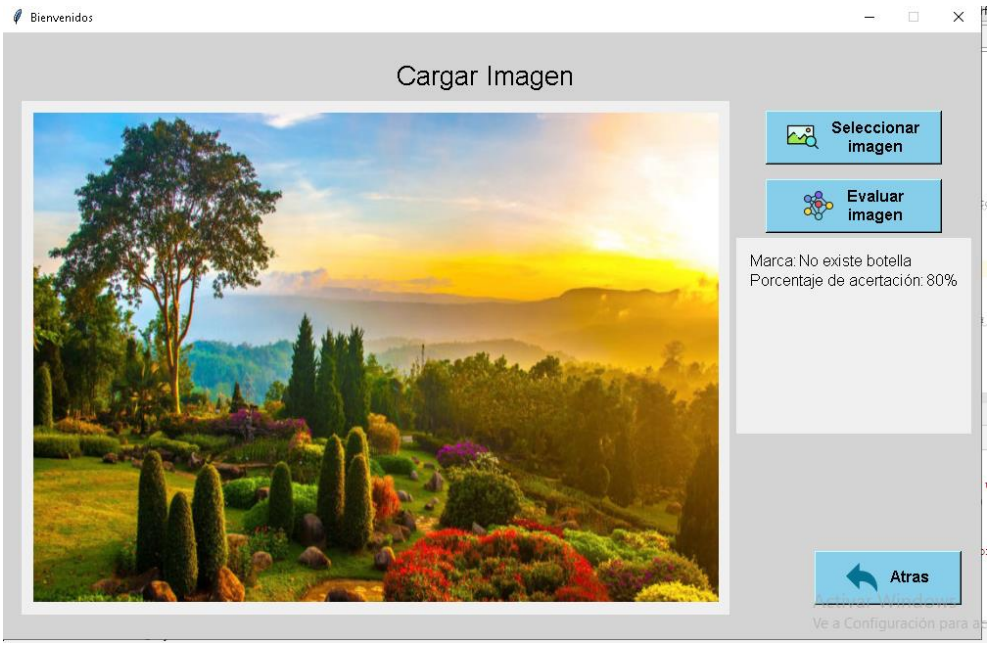


Figura 36: Análisis de una imagen no existente en las categorías de la red neuronal

Para la utilización de la cámara en tiempo real la red necesita del modelo en donde se generará la estructura, las etiquetas para la evaluación de los resultados y los pesos que

permiten a la red ajustarse para que pueda generar predicciones. Luego de esto la red empezara a tomar los datos por medio de la cámara conectada y enviarlos a la red para el análisis y predicción, para posteriormente obtener el mensaje de que tipo de marca es el objeto y el porcentaje de aceptación.

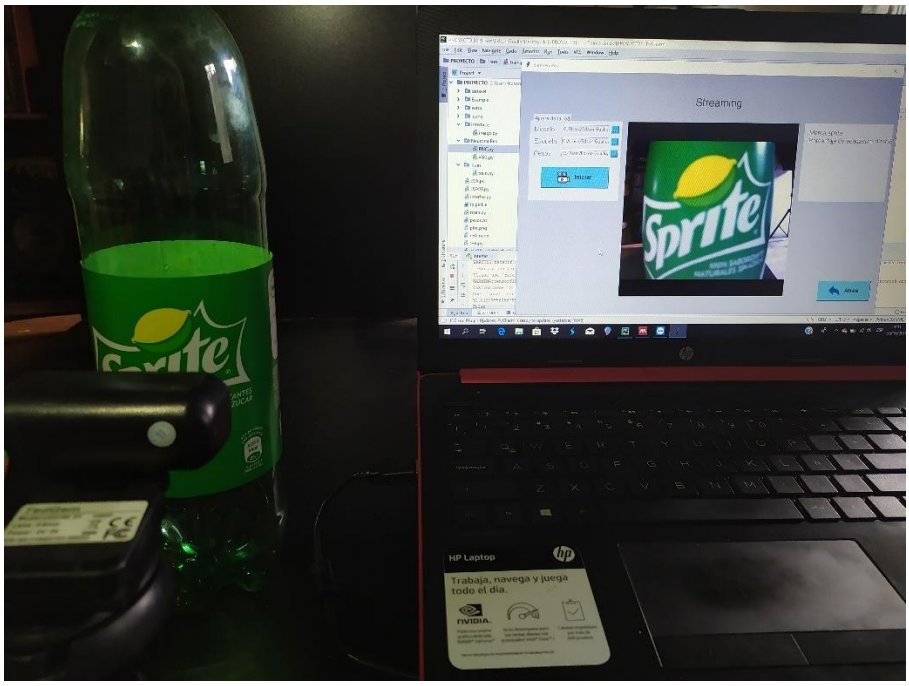


Figura 37: Pantalla de la conexión entre la cámara y el prototipo

Tras la aplicación del prototipo en donde se puede apreciar con claridad como funciona externamente el aplicativo haciendo que a través de una cámara web él envió de datos sea en usb de manera más directa y rápida, constando de una plataforma para su estabilidad.

Figura 38: Prototipo del sistema (Camara y plataforma)

CONCLUSIONES

- Las redes neuronales mantienen su funcionalidad de manera simple, porque al requerir de muchos ejemplos para su entrenamiento, al momento de ejecutarse y ver cómo es su resultado no consume de tanta memoria, lo cual permite a futuro generar avances siendo aplicadas en nuestro día a día.
- Sus aplicaciones son en múltiples áreas. Varios son enfocados a la medicina porque estas observan cosas que el simple ojo humano a través de experimentos no lo consigue, en ciertos casos hasta de incluso predecir enfermedades que se puedan genera a futuro y darle tratamiento antes de que sea demasiado tarde.
- El proyecto al ser un prototipo no consiguió tener toda su funcionalidad con respecto a la red al máximo ya que para poder complementarlo se necesita una cantidad inmensa de datos, los cuales con un correcto tratamiento permite que la red analice y tome sus propias conclusiones, permitiendo que la red analice imágenes o datos completamente nuevos y de un resultado de clasificación acorde a lo ella conoce.
- El dataset óptimo para que la red funcione en favorables condiciones fue de 150 imágenes las cuales consistían de los 3 tipos de marcas y el ultimo que se basa en imágenes aleatorias para que la red reconozca fondos en caso de no existir ningún objeto, los parámetros de entrenamiento fueron con una época de 200 y una taza de aprendizaje optimizable por parte de la librería Tensorflow.
- Luego del entramiento de la red brindo resultados favorables en su reconocimiento para la marca Coca Cola, en el caso de la Sprite su nivel de aceptación fue bajo en comparación a la anterior, mientras que la marca Fanta obtuvo un porcentaje mínimo en referencia a todo el grupo de estudio

RECOMENDACIONES

- Para un mejor funcionamiento del sistema, lo fundamental es que adquieran un gran conjunto de datos, dando así un entrenamiento efectivo, debido a que la red reconocerá cada filtro que contengan las marcas.
- En caso que se quiera colocar más de un filtro es necesario convertir a números binarios, todas las etiquetas a examinar, porque así es como la red puede generar una predicción, caso contrario el entrenamiento seria defectuoso.
- Las diversas perspectivas que se pueda obtener de cada etiqueta, proporcionaran un alta la capacidad de entrenamiento a la red para de predecir la imagen sin necesidad de estar colocándolo con un solo punto de enfoque.
- Si se desea evitar el sobreajuste de una red, lo recomendable es que se tenga un control ya sea teórico de los parámetros a ofrecerle, fijándose que un alto número de épocas con una taza de aprendizaje dada al azar pueden ocasionar que la red aprenda de memoria cada imagen y al ser evaluada esta podría dar un resultado sin ser capaz de realizar una predicción.
- Para programas más desarrollados esto podría hacer que un proyecto deba tener varias cámaras apuntando al mismo objeto y predecir en base a los resultados que arroje cada ángulo.

REFERENCIAS BIBLIOGRÁFICAS

Antona Cortés, C. (2017). *HERRAMIENTAS MODERNAS EN REDES NEURONALES: LA LIBRERÍA KERAS*. Retrieved from https://repositorio.uam.es/bitstream/handle/10486/677854/antona_cortes_carlos_tfg.pdf?sequence=1&isAllowed=y

Arteaga, M. (2018). DESARROLLO DE UN SISTEMA DE VISIÓN ARTIFICIAL PARA DETECTAR AUTOMÓVILES ESTACIONADOS EN LUGARES NO PERMITIDOS TRABAJO.

Cocconi, D., Yuan, R., Mulassano, M., & Ferreyra, D. (2019). Aplicación de una arquitectura de red neuronal para el monitoreo de carga por métodos no invasivos (NILM) utilizando ciclos de activación de artefactos eléctricos en el entrenamiento, (1).

de Benito Gorrón, D. (2018). Detección de voz y música en un corpus a gran escala de eventos de audio.

Durán Suárez, J., Del, A., Torres, R., & Suárez, D. (2017). Redes Neuronales Convolucionales en R Reconocimiento de caracteres escritos a mano Redes Neuronales Convolucionales en R Reconocimiento de caracteres escritos a mano Redes Neuronales Convolucionales en R, 78. Retrieved from http://bibing.us.es/proyectos/abreproy/91338/fichero/TFG+Jaime+Durán+Suárez.pdf

Garrigues Carbó, P. (2018). DISEÑO E IMPLEMENTACIÓN DE UN CLASIFICADOR MEDIANTE REDES NEURONALES PARA UN SISTEMA DE INSPECCIÓN INDUSTRIAL 3D.

Ghojogh, B., & Crowley, M. (2019). The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial, 1–23. Retrieved from http://arxiv.org/abs/1905.12787

Kanaan Izquierdo, S., & Ventura Royo, C. (2016). *Diseño de un sistema de reconocimiento automático de matrículas de vehículos mediante una red neuronal convolucional*. Retrieved from http://openaccess.uoc.edu/webapps/o2/bitstream/10609/52222/7/fnunezsTFM0616memoria.pdf

Kyurkchiev, N., & Markov, S. (2015). Sigmoidal Functions: Some Computational and Modelling Aspects. *Biomath Communications*, *1*(2). https://doi.org/10.11145/j.bmc.2015.03.081

Maricalva, M. (2017). Extracción de atributos faciales mediante redes convolucionales.

Martínez, J., & Mingo, F. (2018). RECONOCIMIENTO DE IMÁGENES MEDIANTE REDES NEURONALES CONVOLUCIONALES. Retrieved from http://oa.upm.es/53050/1/TFG_JAVIER_MARTINEZ_LLAMAS.pdf

Matich, D. J. (2002). Redes Neuronales: Conceptos Básicos y Aplicaciones. *Cátedra: Informática Aplicada a La Ingeniería de Procesos-Orientación I*, 6. Retrieved from https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf

Medina Lescano, G. F. (2017). DESARROLLO DE UNA MÁQUINA ENVASADORA Y TAPADORA DE YOGURT PARA LA FÁBRICA LÁCTEOS SAN FRANCISCO.

Mehryar, M., Rostamizadeh, A., & Talwalkar, A. (2012). *Foundations of Machine Learning*.

Pérez Carrasco, J. A., Serrano Gotarredona, M. del C., Acha Piñero, B., Serrano Gotarredona, M. T., & Linares Barranco, B. (2011). Red neuronal convolucional rápida sin fotograma para el reconocimeinto de dígitos. *XXVI Simposio de La URSI*, (1), 1–4. Retrieved from https://idus.us.es/xmlui/bitstream/handle/11441/79308/RED NEURONAL.pdf?sequence=1&isAllowed=y

Pusiol, P. D. (2014). Redes Convolucionales en Comprensión de Escenas.

Rusell, S. J., & Norvig, P. (2004). *Inteligencia Artificial un Enfoque Moderno* (Pearson Ed). Madrid.

Salas, R. (2005). Redes Neuronales Artificiales, 7. Retrieved from https://s3.amazonaws.com/academia.edu.documents/37429671/Redes_Neuronales_Artificiales.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1556258957&Signature=xSYptS80OJgE%2B4aLrnxZNtCa%2F1U%3D&response-content-disposition=inline%3B filename%3DRedes_Neuronal

Távara Idrogo, J. G. (2019). Modelo de reconocimiento automático de señales de trámsito vehicular mediante aprendizaje profundo de redes neuronales convolucionales, (061), 5–6.

TORRICO RAMOS, F. I. (2017). VALORES SINGULARES DE HANKEL Y REGRESIÓN SOFTMAX PARA DIAGNOSTICO DE SEVERIDAD DE FALLOS EN RODAMIENTOS.

Vilagran, A. (2018). Facial Expression Detection using Convolutional Neural Networks.

ANEXOS

Modulo estructural de la red neuronal

```
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dropout
from keras.layers.core import Dense
from keras import backend as K

class NeuronalRedConvolutional:
    @staticmethod
    def const(width, height, depth, classes):
        red = Sequential() #Se inicia la red para agregar capas
        dimEnt = (height, width, depth) #Son las dimesiones que tendran las imagenes; alto, ancho, y dimensiones las cuales son 3 por la escala RGB
        Chan = -1 #Es para ubicar el orden del canal si channel_last obtendra los datos de imagenes como: (height, width, depth)
        if K.image_data_format() == 'channel_first':
            dimEnt = (depth, height, width) #Es para ubicar el orden del canal si channel_last obtendra los datos de imagenes como: (depth, height, width)
            Chan = 1

        red.add(Conv2D(64, (5,5), padding='same', input_shape=dimEnt))
        red.add(Activation('relu'))
        red.add(BatchNormalization(axis=Chan))
        red.add(MaxPooling2D(pool_size=(3,3)))
        red.add(Dropout(0.25)) #Desconecta neuronas aleatoriamente esto se usa para eliminar la redundancia

        red.add(Conv2D(132, (5, 5), padding='same'))
        red.add(Activation('relu'))
        red.add(BatchNormalization(axis=Chan))
        red.add(Conv2D(132, (5, 5), padding='same'))
```

```
        red.add(Activation('relu'))
        red.add(BatchNormalization(axis=Chan))
        red.add(MaxPooling2D(pool_size=(3,3)))
        red.add(Dropout(0.25))

        red.add(Conv2D(210, (5, 5), padding='same'))
        red.add(Activation('relu'))
        red.add(BatchNormalization(axis=Chan))
        red.add(Conv2D(210, (5, 5), padding='same'))
        red.add(Activation('relu'))
        red.add(BatchNormalization(axis=Chan))
        red.add(MaxPooling2D(pool_size=(3, 3)))
        red.add(Dropout(0.25))

        red.add(Conv2D(132, (5, 5), padding='same'))
        red.add(Activation('relu'))
        red.add(BatchNormalization(axis=Chan))
        red.add(Conv2D(132, (5, 5), padding='same'))
        red.add(Activation('relu'))
        red.add(BatchNormalization(axis=Chan))
        red.add(MaxPooling2D(pool_size=(3, 3)))
        red.add(Dropout(0.25))

        red.add(Flatten()) #Sirve para unir las capas de convoluciones
con las densas
        red.add(Dense(1200))
        red.add(Activation('relu'))
        red.add(BatchNormalization())
        red.add(Dropout(0.25))

        red.add(Dense(classes))
        red.add(Activation('softmax'))

        return red
```

Menú principal de la aplicación

```
from tkinter import *
from tkinter import messagebox
import tkinter.filedialog as tkf
import tkinter as tk
from Train.train import RTrain as rt
import os.path as path
from Interface.image import Image as img
import cv2
from PIL import Image, ImageTk
import threading as thr
import time

class Menu:
    def __init__(self):
        self.window = Tk()
        self.window.title("Bienvenidos")
        # Tamaño de la ventana
        self.window.geometry('520x480+440+150')
        # Cambiar Icono
        # window.iconbitmap("form.ico")
        self.window.config(bg='#D3D3D3')
        self.oimg = ""
        self.smod = ''
        self.setq = ''
        self.spes = ''
        self.imga = ImageTk.PhotoImage(file='Icono/atras.png')
        self.window.protocol('WM_DELETE_WINDOW', self.cerrar)

        # Etiquetas
        lbl = Label(self.window, text="Menú", font=("Arial Bold", 50),
bg='#D3D3D3')
        lbl.place(x=175, y=50)
        # Botones
        btn = Button(self.window, text="Entrenar Red", bg='#87CEEB',
font='Helvetica 12 bold', command=self.abrirventana2)
        btn.place(x=80, y=170, width=180, height=55)
        btn2 = Button(self.window, text="   Streaming", bg='#87CEEB',
font='Helvetica 12 bold', command=self.abrirventana3)
        btn2.place(x=280, y=170, width=180, height=55)
```

```
        btn3 = Button(self.window, text="Cargar Imagen", bg='#87CEEB',
font='Helvetica 12 bold', command=self.abrirventana4)
        btn3.place(x=180, y=250, width=180, height=55)
        self.window.resizable(0, 0)
        self.window.mainloop()
#Entrenar red
    def abrirventana2(self):
        self.window.withdraw()
        self.window2 = tk.Toplevel()
        self.window2.geometry('800x620+300+60')
        self.window2.config(bg='#D3D3D3')
        self.window2.resizable(0, 0)

        self.window2.protocol('WM_DELETE_WINDOW', self.cerrar2)

        lbl = Label(self.window2, text="Entrenar Red", font=("Arial
Bold", 20), bg='#D3D3D3')
        lbl.place(x=320, y=50)

        pld = PanedWindow(self.window2)
        pld.place(x=14, y=90, width=253, height=150)

        lbl = Label(self.window2, text="Dir. del dataset:", font=("Arial
Bold", 13))
        lbl.place(x=20, y=115)
        self.dc = Entry(self.window2, width=15, state='readonly')
        self.dc.place(x=140, y=117)
        btn = Button(self.window2, text="...", bg='#87CEEB',
font='Helvetica 12 bold', command= self.dirt)
        btn.place(x=240, y=117, width=20, height=20)

        lbl = Label(self.window2, text="Epocas:", font=("Arial Bold",
13))
        lbl.place(x=20, y=145)
        self.ep = Entry(self.window2, width=20)
        self.ep.place(x=140, y=147)
        self.ep.bind('<KeyRelease>', self.kb)
        self.ep.focus()
        self.ep.bind('<KeyPress>', self.kb)
        self.ep.focus()
```

```
        self.pl = PanedWindow(self.window2)
        self.pl.place(x=270, y=90, width=500, height=400)
        scr = Scrollbar(self.pl)
        scr.pack(side=RIGHT, fill = Y)

        self.imgt = ImageTk.PhotoImage(file='Icono/trai.png')
        self.imgr = ImageTk.PhotoImage(file='Icono/res.png')

        btn = Button(self.window2, text="   Entrenar", image=self.imgt,
compound=LEFT, bg='#87CEEB', font='Helvetica 12 bold', command=lambda
:thr.Thread(target=self.entRed).start())
        btn.place(x=60, y=180, width=180, height=45)

        btn = Button(self.window2, text="   Resultados del\n
entrenamiento", image=self.imgr, compound=LEFT, bg='#87CEEB',
font='Helvetica 12 bold', command=self.abrirventana5)
        btn.place(x=25, y=433, width=240, height=55)

        btn = Button(self.window2, text="   Atras", image=self.imga,
compound = LEFT, bg='#87CEEB', font='Helvetica 12 bold',
command=self.cerrar2)
        btn.place(x=620, y=500, width=150, height=50)

    def kb(self,event):
        p = self.ep.get()
        if not str.isnumeric(p) or (len(p) >= 4) or (int(p) >= 300):
            p = p[:-1]
            self.ep.delete(0, END)
            self.ep.insert(0, p)

    def dirt(self):
        p = tkf.askdirectory()
        self.dc.configure(state='normal')
        self.dc.delete(0, END)
        self.dc.insert(0, p)
        self.dc.configure(state='readonly')

    def entRed(self):
        if self.ep.get() != '' and self.dc.get() != '':
```

```
            rt.redTrain(int(self.ep.get()), self.dc.get(), pl=self.pl)
        else:
            messagebox.showinfo('Alerta', 'La ubicacion de la carpeta no
esta seleccionada o las epocas no estan declaradas')
#Streaming
    def abrirventana3(self):
        self.window.withdraw()
        self.window3 = tk.Toplevel()
        self.window3.geometry('1000x620+200+50')
        self.window3.config(bg='#D3D3D3')

        self.window3.resizable(0, 0)

        self.window3.protocol('WM_DELETE_WINDOW', self.cerrar3)

        lbl = Label(self.window3, text="Streaming", font=("Arial Bold",
20), bg='#D3D3D3')
        lbl.place(x=450, y=50)

        pl2 = PanedWindow(self.window3)
        pl2.place(x=26, y=117, width=240, height=200)

        lbl = Label(self.window3, text="Ajuste de la red", font=("Arial
Bold", 10))
        lbl.place(x=34, y=100)

        lbl = Label(self.window3, text="Modelo:", font=("Arial Bold",
13))
        lbl.place(x=35, y=125)
        self.mod = Entry(self.window3, width=20, state='readonly')
        self.mod.place(x=110, y=127)
        btn = Button(self.window3, text="...", bg='#87CEEB',
font='Helvetica 12 bold', command= lambda :self.cam(tip='mod'))
        btn.place(x=238, y=127, width=20, height=20)

        lbl = Label(self.window3, text="Etiqueta:", font=("Arial Bold",
13))
        lbl.place(x=35, y=155)
        self.etq = Entry(self.window3, width=20, state='disabled')
        self.etq.place(x=110, y=157)
```

```
        btn = Button(self.window3, text="...", bg='#87CEEB',
font='Helvetica 12 bold', command=lambda :self.cam(tip='etq'))
        btn.place(x=238, y=157, width=20, height=20)

        lbl = Label(self.window3, text="Pesos:", font=("Arial Bold", 13))
        lbl.place(x=35, y=185)
        self.pes = Entry(self.window3, width=20, state='disabled')
        self.pes.place(x=110, y=187)
        btn = Button(self.window3, text="...", bg='#87CEEB',
font='Helvetica 12 bold', command=lambda :self.cam(tip='pes'))
        btn.place(x=238, y=187, width=20, height=20)

        #self.stop_threads = False
        #self.th = thr.Thread(target=self.run, args =(self, lambda :
self.stop_threads, )) #, args =(lambda : self.stop_threads, )

        self.imgr = ImageTk.PhotoImage(file='Icono/rec.png')
        btn = Button(self.window3, text="   Iniciar",image=self.imgr,
compound=LEFT, bg='#87CEEB', font='Helvetica 12 bold', command=self.run)
        btn.place(x=58, y=230, width=180, height=55)

        btn = Button(self.window3, text="   Atras", image=self.imga,
compound = LEFT, bg='#87CEEB', font='Helvetica 12 bold',
command=self.cerrar3)
        btn.place(x=830, y=530, width=150, height=55)

        self.pl2 = PanedWindow(self.window3)
        self.pl2.place(x=280, y=117, width=455, height=455)
        self.lblimg = Label(self.pl2)
        self.lblimg.pack()

        self.pl3 = PanedWindow(self.window3)
        self.pl3.place(x=750, y=117, width=240, height=200)

        lbl = Label(self.pl3, text='Marca:', font=("Arial Bold", 12))
        lbl.place(x=10, y=10)
        self.lblmc = Label(self.pl3, text='', font=("Arial Bold", 12))
        self.lblmc.place(x=60, y=10)
        lbl = Label(self.pl3, text='Porcentaje de acertación:',
font=("Arial Bold", 12))
```

```
        lbl.place(x=10, y=30)
        self.lblpa = Label(self.pl3, text='', font=("Arial Bold", 12))
        self.lblpa.place(x=190, y=30)

    def cam(self, tip='run'):
        if tip == 'mod':
            p = tkf.askopenfilename(title="Seleccionar archivo",
filetypes=(("model files", ".model"),))
            self.mod.configure(state='normal')
            self.mod.delete(0,END)
            self.smod = p
            self.mod.insert(0,p)
            self.mod.configure(state='readonly')
        if tip == 'etq':
            p = tkf.askopenfilename(title="Seleccionar archivo",
filetypes=(("model files", ".pickle"),))
            self.etq.configure(state='normal')
            self.etq.delete(0,END)
            self.setq = p
            self.etq.insert(0,p)
            self.etq.configure(state='readonly')
        if tip == 'pes':
            p = tkf.askopenfilename(title="Seleccionar archivo",
filetypes=(("model files", ".h5"),))
            self.pes.configure(state='normal')
            self.pes.delete(0,END)
            self.spes = p
            self.pes.insert(0,p)
            self.pes.configure(state='readonly')

    def run(self, tip='run'):
        self.stop_threads = False
        if tip == 'run':
            if self.smod != '' and self.setq != '' and self.spes != '':
                if path.exists(self.setq) and path.exists(self.smod) and
path.exists(self.spes):
                    self.th = thr.Thread(target=img.live, args =(lambda:
self.stop_threads,self.smod,self.setq,self.spes,self.lblimg,self.lblmc,se
lf.lblpa))
                    #self.th.setDaemon(True)
```

```
                    self.th.start()
                    #self.stop_threads = True
                    #time.sleep(2)
                    #stop_threads = True
                    #self.th.join()

#img.live(dMod=self.smod,dEtq=self.setq,dWht=self.spes,pl=self.pl2,lbl=se
lf.lblimg,lblmc=self.lblmc,lblpa=self.lblpa)
            else:
                messagebox.showinfo('Alerta', 'No ha seleccionado los
archivos para la red')
        if tip == 'stp':
            self.stop_threads = True
            #self.th.join(3.0)
#Cargar Imagen
    def abrirventana4(self):
        self.window.withdraw()
        self.window4 = tk.Toplevel()
        self.window4.geometry('1000x620+200+50')
        self.window4.config(bg='#D3D3D3')

        self.window4.resizable(0, 0)

        self.window4.protocol('WM_DELETE_WINDOW', self.cerrar4)

        lbl = Label(self.window4, text="Cargar Imagen", font=("Arial
Bold", 20), bg='#D3D3D3')
        lbl.place(x=400, y=25)
        self.imgs = ImageTk.PhotoImage(file='Icono/sel.png')
        btn = Button(self.window4, text="   Seleccionar\n   imagen",
image=self.imgs, compound=LEFT, bg='#87CEEB', font='Helvetica 12 bold',
command=self.cImagen)
        btn.place(x=780, y=80, width=180, height=55)
        self.imgr = ImageTk.PhotoImage(file='Icono/red.png')
        btn = Button(self.window4, text="   Evaluar\n
imagen",image=self.imgr, compound=LEFT, bg='#87CEEB', font='Helvetica 12
bold', command=self.EImg)
        btn.place(x=780, y=150, width=180, height=55)
        self.pl0 = PanedWindow(self.window4)
        self.pl0.place(x=20, y=70, width=723, height=525)
```

```
        self.pn = PanedWindow(self.window4)
        self.pn.place(x=750, y=210, width=240, height=200)
        #scr = Scrollbar(self.pn)
        #scr.pack(side=RIGHT, fill = Y)
        lbl = Label(self.pn, text='Marca:', font=("Arial Bold", 12))
        lbl.place(x=10, y=10)
        self.lblpl = Label(self.pn, text='', font=("Arial Bold", 12))
        self.lblpl.place(x=60, y=10)
        lbl = Label(self.pn, text='Porcentaje de acertación:',
font=("Arial Bold", 12))
        lbl.place(x=10, y=30)
        self.lblpt = Label(self.pn, text='', font=("Arial Bold", 12))
        self.lblpt.place(x=190, y=30)
        btn = Button(self.window4, text="   Atras", image=self.imga,
compound = LEFT, bg='#87CEEB', font='Helvetica 12 bold',
command=self.cerrar4)
        btn.place(x=830, y=530, width=150, height=55)

    def cImagen(self):
        p = tkf.askopenfilename(title = "Seleccionar archivo" , filetypes
= (( "jpeg files" , ".jpg" ),))
        if p != '':
            self.oimg = p
            img = None
            val = False
            if path.exists(p):
                val = True
                p = path.normpath(p)
            if val == True:
                img = cv2.imread(p)
                img = cv2.resize(img, (700,500))
                img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                img = Image.fromarray(img)
                img = ImageTk.PhotoImage(img)
                lbl = Label(self.window4)
                lbl.imgtk = img
                lbl.configure(image = img)
                lbl.place(x=30, y=80)

    def EImg(self):
```

```
        if path.exists(self.oimg):
            dImg = path.normpath(self.oimg)
            dMod = 'red.model'
            dWht = 'pesos.h5'
            dEtq = 'lb.pickle'
            result = img.Img(dImg=dImg, dMod=dMod, dEtq=dEtq, dWht=dWht)
            self.lblpl.config(text=result.split(',')[0])
            self.lblpt.config(text=result.split(',')[1])
            #self.lblpl = Label(self.pn, text=result.split(',')[0],
font=("Arial Bold", 12), bg='#D3D3D3')
            #self.lblpl.place(x=10, y=10)
        else:
            messagebox.showinfo('Alerta', 'No se ha seleccionado ninguna
imagen')
#Estadistica
    def abrirventana5(self):
        self.window2.withdraw()

        self.window.withdraw()
        self.window5 = tk.Toplevel()
        self.window5.geometry('700x620+380+50')
        self.window5.config(bg='#D3D3D3')
        self.window5.title ('Estadistica')

        self.window5.resizable(0, 0)

        self.window5.protocol('WM_DELETE_WINDOW', self.cerrar5)

        lbl = Label(self.window5, text="Graficos Estadisticos",
font=("Arial Bold", 20), bg='#D3D3D3')
        lbl.place(x=200, y=20)
        self.pl = PanedWindow(self.window5)
        self.pl.place(x=20, y=60, width=663, height=495)
        #img = Image.open('Icono/reportar.png')
        self.imgb = ImageTk.PhotoImage(file='Icono/reportar.png')
        btn = Button(self.window5, text="Generar", image=self.imgb,
compound = LEFT, bg='#87CEEB', font='Helvetica 12 bold',
command=self.genimg)
        btn.place(x=570, y=25, width=120, height=50)
        self.lblins = Label(self.window5)
```

```
        btn = Button(self.window5, text="   Atras", image=self.imga,
compound = LEFT, bg='#87CEEB', font='Helvetica 12 bold',
command=self.cerrar5)
        btn.place(x=530, y=563, width=150, height=55)

    def genimg(self):
        val = False
        if path.exists("plot.png"):
            val = True
        if val:
            img = cv2.imread("plot.png")
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = Image.fromarray(img)
            img = ImageTk.PhotoImage(img)
        self.lblins.imgtk = img
        self.lblins.configure(image=img)
        self.lblins.place(x=30, y=65)

    def cerrar(self):
        exit(0)

    def cerrar2(self):
        self.window2.withdraw()
        self.__init__()

    def cerrar3(self):
        #img.b = False
        self.run(tip='stp')
        self.window3.withdraw()
        self.__init__()

    def cerrar4(self):
        self.window4.withdraw()
        self.__init__()

    def cerrar5(self):
        self.window5.withdraw()
        self.abrirventana2()

c = Menu()
```

Módulo de configuración de imágenes y entrenamiento de la red

```
import matplotlib
matplotlib.use('Agg') #Se usa para que las figuras puedan guardar en el
fondo
from tkinter import *
from keras.preprocessing.image import ImageDataGenerator #Sirve para el
aumento de datos que se les aplica transformaciones para tener muchas mas
cantidades
from keras.optimizers import Adam #Es un optimizador que se acopla a
mejor medida dependiendo del resultado de la epoca
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import LabelBinarizer #Transforma una cadena
de etiqueta en un entero y viceversa
from sklearn.model_selection import train_test_split#Crea divisiones de
entrenamiento y prueba
from NeuronalRed.RNC import NeuronalRedConvolutional as RNC
import matplotlib.pyplot as plt
from imutils import paths
import numpy as np
import random
import pickle
import cv2
import os
import threading as thr


class RTrain:
    @staticmethod
    def redTrain(epoc, dir, pl):
        lbl = Label(pl, text="Entro")
        lbl.pack()
        EPOCHS = epoc
        Lr = 1e-3
        Bs = 10
        IMG_DIM = (200,200,3) #Antes de 120x120

        data = []
        labels = []

        lbl = Label(pl, text="[INFO] Cargando imagenes")
        lbl.pack()

        imagePaths = sorted(list(paths.list_images(dir)))  # Captura la
rutas de las imagenes para ser tratadas
        random.seed(42)  # Crea una nueva semilla de aleatoriedad
        random.shuffle(imagePaths)  # Baraja las imagenes que se han
encontrado
        lbl = Label(pl, text="[INFO] Proceso exitoso...")
        lbl.pack()

        lbl = Label(pl, text="[INFO] Tratando imagenes")
        lbl.pack()
        for imagePath in imagePaths:
            image = cv2.imread(imagePath)  # Se carga la imagen
            image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
            image = cv2.resize(image, (IMG_DIM[0], IMG_DIM[1]))  # Se
redimensiona la imagen actual por los pixeles dados
            image = img_to_array(image)  # Permite convertir una imagen
en matriz
```

```
            data.append(image)  # Se agrega la matriz a una lista llamada
data

            label = imagePath.split(os.path.sep)[-2]  # Se separa el
nombre del directorio ya sea el nombre de la etiqueta
            labels.append(label)  # Se agrega a la lista el nombre del
pokemon ya extraido anteriormente
        lbl = Label(pl, text="[INFO] Proceso exitoso...")
        lbl.pack()

        lbl = Label(pl, text="[INFO] [INFO] Redimensionando")
        lbl.pack()
        data = np.array(data,dtype='float') / 255.0  # Se genera la
division de la intesidad de los pixeles para que en vez de 255 sea el
maximo solo llegue a 1
        labels = np.array(labels)  # Se genera un array de los nombres de
pokemons
        lbl = Label(pl, text="[INFO] data matrix:
{:.2f}MB".format(data.nbytes / (1024 * 1000.0)))
        lbl.pack()

        lbl = Label(pl, text="[INFO] Binarizando las etiquetas,
dividiendo los datos y modificando imagenes para alimentar a la red")
        lbl.pack()
        lb = LabelBinarizer()  # Se instancia la binarizacion
        labels = lb.fit_transform(labels)  # Combierte cada etiqueta en
un numero binario para que la maquina pueda asociarlo
        #   Division de los datos de entrenamiento y de validacion,
tomando el 80% para entrenamiento y el 20% restante para la validacion
        (trainX, testX, trainY, testY) = np.array(train_test_split(data,
labels, test_size=0.2, random_state=42))
        aug = ImageDataGenerator(rotation_range=25,
width_shift_range=0.1, height_shift_range=0.1, shear_range=0.2,
                                 zoom_range=0.2, horizontal_flip=True,
fill_mode='nearest')
        lbl = Label(pl, text="[INFO] Proceso exitoso...")
        lbl.pack()

        lbl = Label(pl, text="[INFO] Compilando modelo...")
        lbl.pack()
        model = RNC.const(width=IMG_DIM[1], height=IMG_DIM[0],
depth=IMG_DIM[2],classes=len(lb.classes_))
        opt = Adam(lr=Lr, decay=Lr / EPOCHS)
        model.compile(loss='categorical_crossentropy', optimizer=opt,
metrics=["accuracy"])
        lbl = Label(pl, text="[INFO] Proceso exitoso...")
        lbl.pack()

        #Hasta aqui, Freezeo de pantalla porque esta en segundo plano
        lbl = Label(pl, text="[INFO] Entrenando al modelo, por favor
espere...")
        lbl.pack()
        H = model.fit_generator(aug.flow(trainX, trainY, batch_size=Bs),
validation_data=(testX, testY),
                                steps_per_epoch=len(trainX) // Bs,
epochs=EPOCHS, verbose=1)
        lbl = Label(pl, text="[INFO] Entrenamiento exitoso...")
        lbl.pack()

        lbl = Label(pl, text="[INFO] Guardando modelo y pesos")
```

```
        lbl.pack()
        model.save('red.model')
        model.save_weights('pesos.h5')
        lbl = Label(pl, text="[INFO] Guardado exitoso...")
        lbl.pack()

        lbl = Label(pl, text="[INFO] Serializando etiquetas y
guadandolas")
        lbl.pack()
        f = open('lb.pickle', "wb")
        f.write(pickle.dumps(lb))
        f.close()
        lbl = Label(pl, text="[INFO] Proceso exitoso...")
        lbl.pack()

        lbl = Label(pl, text="[INFO] Generando graficos para luego
guardarlos")
        lbl.pack()
        plt.style.use("ggplot")
        plt.figure()
        N = EPOCHS
        plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
        plt.plot(np.arange(0, N), H.history["val_loss"],
label="val_loss")
        plt.plot(np.arange(0, N), H.history["acc"], label="train_acc")
        plt.plot(np.arange(0, N), H.history["val_acc"], label="val_acc")
        plt.title("Training Loss and Accuracy")
        plt.xlabel("Epoch #")
        plt.ylabel("Loss/Accuracy")
        plt.legend(loc="upper left")
        plt.savefig('plot')  # Se guardaran los resultados como una
imagen
        lbl = Label(pl, text="[INFO] Proceso exitoso...")
        lbl.pack()
```

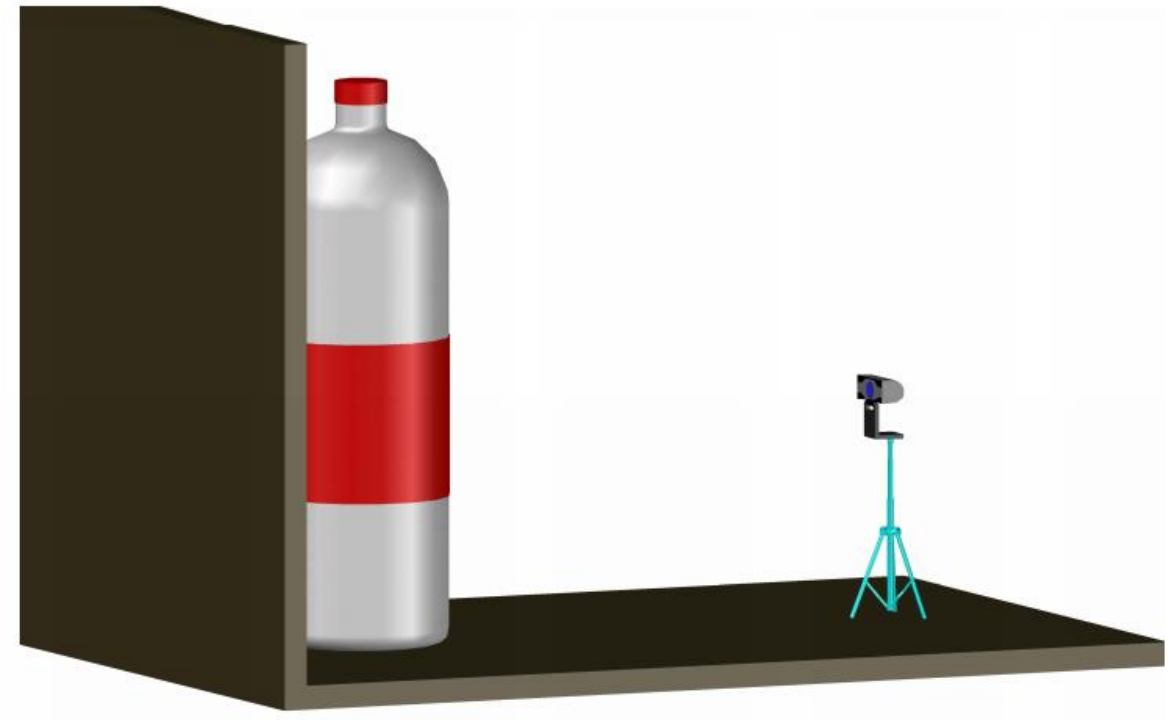


Figura 39: Modelado del prototipo en 3D

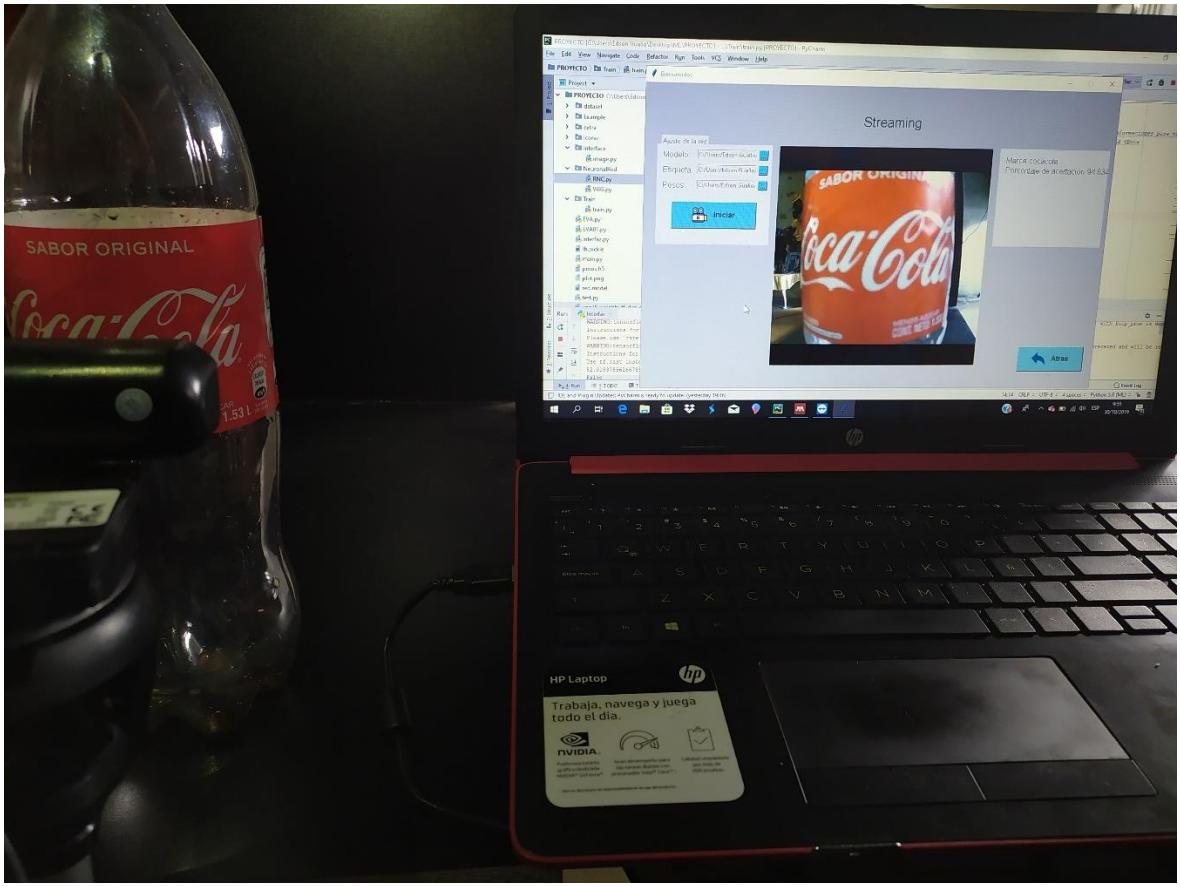


Figura 40: Funcionamiento de sistema y prototipo con la marca Coca Cola

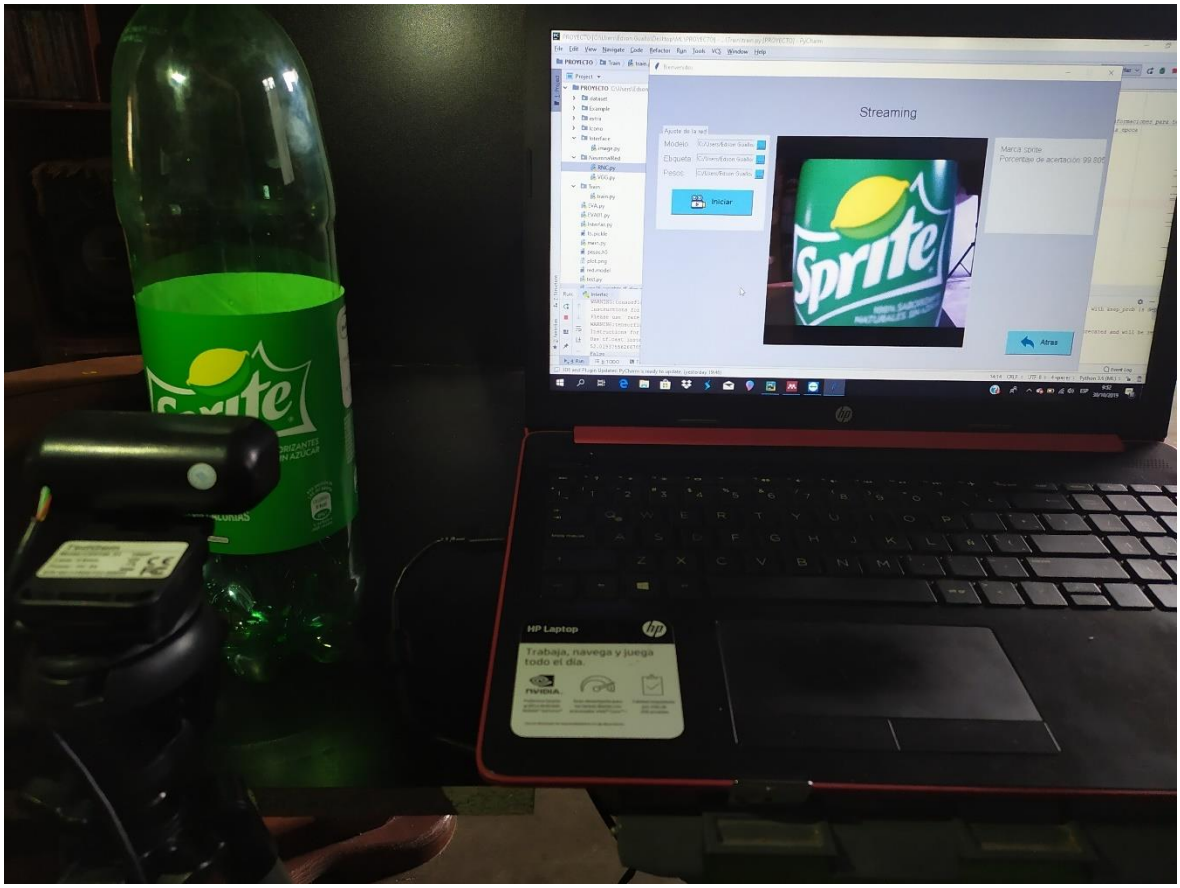


Figura 41: Funcionamiento del sistema y el prototipo de la marca Sprite

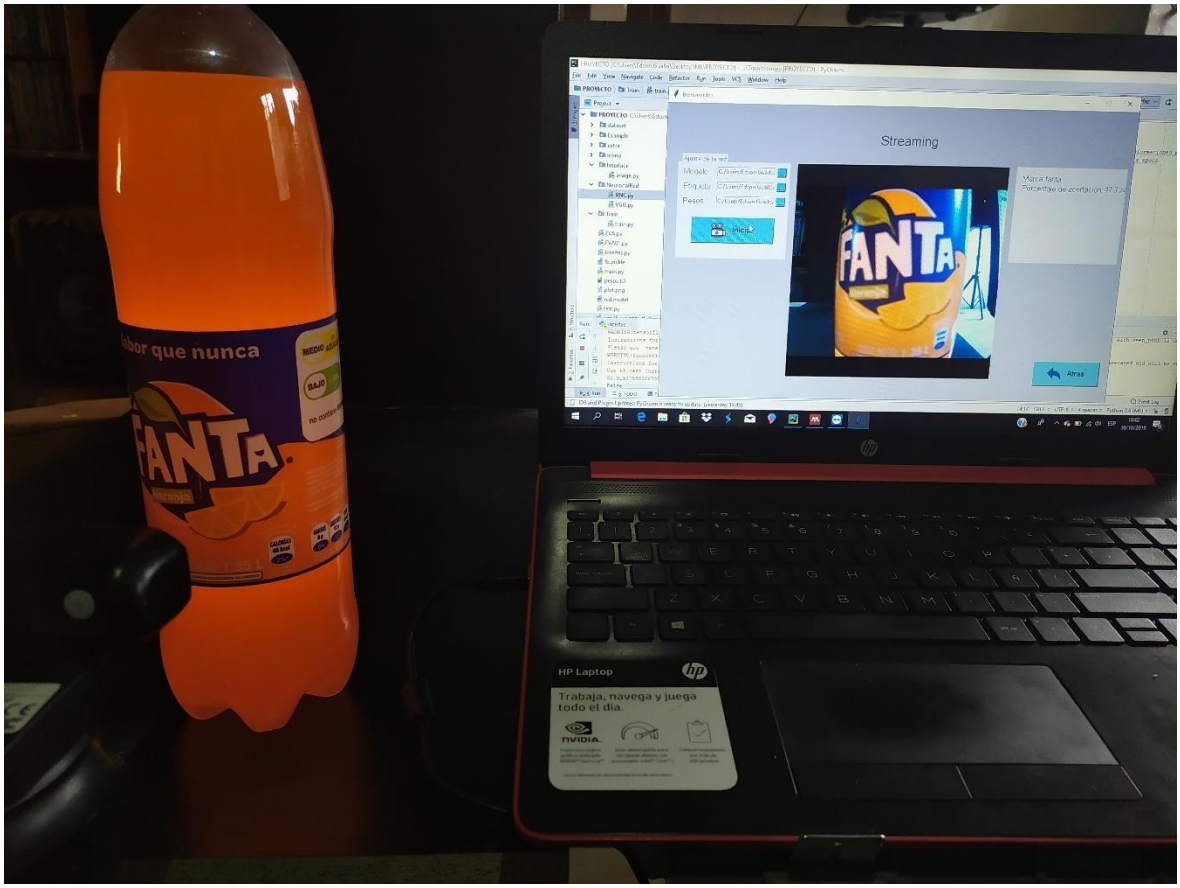


Figura 42: Funcionamiento del sistema y prototipo de la marca Fanta

Redes Convolucionales

INFORME DE ORIGINALIDAD

1%	0%	0%	1%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Politécnico Colombiano Jaime Isaza Cadavid Trabajo del estudiante	1%

Excluir citas	Activo	Excluir coincidencias	< 50 words
Excluir bibliografía	Activo		