“Implementación de un sistema de clasificación binaria basado en el uso de servicios web cognitivos y redes neuronales profundas”

Chancusig Casa, Cristian Alexander y Tumbaco Casa, Sergio David

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica, Automatización y Control

Ing. Alulema Flores, Darwin Omar, PhD.

18 de julio del 2022

COPYLEAKS

Capitulo 1 - 6 Trabajo de Titulación_Chancusig_Tumbaco.pdf

Scanned on: 23:4 July 17, 2022 UTC

DARWIN OMAR ALULEMA FLORES

2.2%	22	31904
Overall Similarity Score	Results Found	Total Words in Text

Identical Words	305
Words with Minor Changes	238
Paraphrased Words	155
Omitted Words	0

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Certificación

Certifico que el trabajo de titulación: "**Implementación de un sistema de clasificación binaria basado en el uso de servicios web cognitivos y redes neuronales profundas**" fue realizado por los señores **Chancusig Casa, Cristian Alexander y Tumbaco Casa, Sergio David;** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Sangolquí, 18 de julio del 2022

Firma:

Firmado electrónicamente por:
DARWIN OMAR ALULEMA FLORES

.....

Dr. Alulema Flores, Darwin Omar

C. C: 1002493334

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Responsabilidad de Autoría

Nosotros, **Chancusig Casa, Cristian Alexander y Tumbaco Casa, Sergio David**, con cédulas de ciudadanía n° 1725903924 y 1726453887, declaramos que el contenido, ideas y criterios del trabajo de titulación: **"Implementación de un sistema de clasificación binaria basado en el uso de servicios web cognitivos y redes neuronales profundas"**, es nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 18 de julio del 2022

Firma

Chancusig Casa, Cristian Alexander
C.C.: 1725903924

Tumbaco Casa, Sergio David
C.C.: 1726453887

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica, Automatización y Control

Autorización de Publicación

Nosotros **Chancusig Casa, Cristian Alexander y Tumbaco Casa, Sergio David**, con cédulas de ciudadanía n° 1725903924 y 1726453887, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **"Implementación de un sistema de clasificación binaria basado en el uso de servicios web cognitivos y redes neuronales profundas"**, en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi/nuestra responsabilidad.

Sangolquí, 18 de julio del 2022

Firma

Chancusig Casa, Cristian Alexander
C.C.: 1725903924

Tumbaco Casa, Sergio David
C.C.: 1726453887

Dedicatoria

Este trabajo se lo dedico a mis padres, por haberme apoyado cada día para alcanzar mis objetivos, muchos de mis logros se los debo a ustedes, incluido este, quienes con su gran paciencia, amor y sabiduría han sabido guiarme en cada momento de mi vida. A mi hermano Darío, quien ha sabido entenderme en los momentos más complicados de mi vida, y me ha brindado su apoyo para poder seguir adelante.

Cristian Alexander Chancusig Casa

Dedicatoria

Dedico este trabajo final a mis padres, pues son el soporte en mi vida, bajo su guía, apoyo y sacrificio me han brindado oportunidades para mi superación y con ellos hoy culmino mi carrera universitaria. A mi hermano Paul, por ser siempre una fuente de inspiración y un ejemplo para mi vida. A mi novia Karen, por el apoyo y confianza brindada en toda mi carrera.

Sergio David Tumbaco Casa

Agradecimiento

Agradezco a Dios, por la vida que me ha regalado, por haber protegido a mi familia y a mí persona, además por darme la fortaleza de no rendirme por más difícil que haya sido el momento en el transcurso de mi carrera universitaria.

Gracias a mis padres, por el amor y comprensión que me han dado, por los grandes consejos que he recibido en los momentos más complicados de mi vida, y también por inculcarme grandes valores los cuales me han formado como una persona de bien.

Gracias a mi hermano, Dario, por el apoyo moral que me ha brindado, y motivarme a seguir superando los diversos obstáculos que se presentan.

Gracias a mis amigos, con quienes he compartido grandes experiencias, y han hecho que este camino sea mucho más alegre.

A mi primo y compañero de tesis, David, quien fue una gran ayuda para el desarrollo y la finalización de este trabajo.

A nuestro tuto el Ing. Darwin Alulema, quien confió en nosotros para sacar adelante el trabajo, y que con su experiencia y enseñanzas supo guiarnos en la elaboración del presente proyecto.

Finalmente agradezco a la Universidad de las Fuerzas Armadas ESPE y a sus docentes, quienes compartieron sus conocimientos conmigo.

Cristian Alexander Chancusig Casa

Agradecimiento

Agradezco a Dios por la vida, por haberme guiado y por su protección durante mis estudios. A mis padres, hermano, y novia por ser el soporte en los momentos de dificultad que se presentaron en mi vida universitaria. A la universidad, mis docentes, amigos y compañeros que me han aportado sus conocimientos para poder crecer de forma profesional. A mi primo y compañero de tesis Cristian, quien supo ser un aporte fundamental en el desarrollo de este trabajo. Finalmente agradezco a mi tutor Ing. Darwin Alulema por ser el guía en la elaboración de este proyecto.

Sergio David Tumbaco Casa

Índice de Contenido

Índice de Tablas

Índice de Figuras

Resumen

Los sistemas basados en la visión por computador e inteligencia artificial resultan una alternativa atractiva dentro de procesos con tareas de inspección, y que se realizan de manera repetitiva, buscando suplir la función del ojo humano. Estas herramientas han sido empeladas en sistemas de clasificación de objetos específicos, sin embargo, es relevante poder ampliar su funcionalidad brindando la capacidad de aprendizaje, y con ello poder clasificar múltiples objetos empleando una misma máquina. Bajo estas consideraciones, se propone un prototipo de un sistema clasificador de objetos basado en el servicio web cognitivo de Microsoft Azure y su API Computer Vision, y el uso de transfer learning y ResNet 50 como una red preentrenada. El servicio en la nube permite al sistema la identificación y etiquetado del contenido de imágenes, mientras que el servicio local basado en la red neuronal permite la generación de modelos de clasificación para aquellos objetos no identificados o identificados de manera incorrecta por el primer servicio. La arquitectura del sistema consta de tres capas, una capa física contenedora de la estructura mecánica y electrónica, una capa lógica contenedora de ambos servicios y una capa de aplicación subdividida en un Backend para la integración de los servicios, y un Frontend para la integración de las interfaces de supervisión y control. Finalmente, el sistema se ha evaluado mediante pruebas de funcionalidad y métricas de desempeño de modelos de clasificación, así como con pruebas de carga y usabilidad, obteniendo buenos resultados.

Palabras clave: Azure, Computer Vision, ResNet 50, transfer learning, Django.

Abstract

Systems based on computer vision and artificial intelligence are an attractive alternative within processes with inspection tasks, and that are carried out repetitively, seeking to replace the function of the human eye. These tools have been used in classification systems for specific objects, however, it is relevant to be able to extend their functionality by providing learning capacity, and thus be able to classify multiple objects using the same machine. Under these considerations, a prototype of an object classifier system based on the Microsoft Azure cognitive web service and its Computer Vision API, and the use of transfer learning and ResNet 50 as a pretrained network, is proposed. The cloud service allows the system to identify and label the content of images, while the local service based on the neural network allows the generation of classification models for those objects not identified or incorrectly identified by the first service. The system architecture consists of three layers, a physical layer that contains the mechanical and electronic structure, a logical layer that contains both services, and an application layer subdivided into a Backend for the integration of services, and a Frontend for the integration of services. monitoring and control interfaces. Finally, the system has been evaluated through functionality tests and performance metrics of classification models, as well as load and usability tests, obtaining good results.

Key words: Azure, Computer Vision, ResNet 50, transfer learning, Django.

Capítulo 1: Marco Metodológico

Antecedentes

La visión en los seres humanos es uno de sus principales sentidos, por medio de ella recibe más del 95% de información de su entorno, brindando una percepción de forma de los objetos a su alrededor (Aguila Antonio Zárate et al., 2009), permitiéndole realizar diferentes procesos, ya sean de identificación, selección, clasificación o de movimiento. Pretendiendo simular el sentido del ojo humano, nace en los años 60 la visión artificial o visión por computadora, con la idea básica de conectar una cámara a un computador, la cual capte información del entorno y la envíe a un procesador para poder ser tratada y utilizada a beneficio. Así también, en busca de brindar capacidades de razonamiento, aprendizaje y planificación a una máquina, nace la inteligencia artificial, la cual es la base a partir de la que se pretende que un computador imite procesos del cerebro humano. Los sistemas de visión por computador junto con la inteligencia artificial generan una herramienta que permite la adquisición, procesamiento y análisis de imágenes, extrayendo de ellas información como la forma y color de los objetos, proporcionando así a la máquina la capacidad de ver e interactuar con su entorno. Debido a estas características, la visión artificial fue ganando espacio en procesos de agricultura, alimentación, control de calidad, seguridad, medicina, transporte, robótica y más.

Dentro del ámbito industrial, las operaciones que tienden a llevar un control de calidad de producto imitan los procesos humanos cotidianos que se realizan al adquirir dicho producto, siendo estas operaciones parte de un sistema clasificador. Tareas de selección y clasificación en las que se requiere separar objetos según características específicas, son comunes dentro de una empresa, por lo que, cumplir con tal misión requiere de un gran personal. En busca de cubrir esta necesidad, el uso de tecnología de visión por computador e inteligencia artificial pretende suplir la necesidad humana para procesos iterativos, mejorando el resultado en

términos de precisión y tiempo. La función dada a un sistema clasificador puede ser extenso, en el trabajo de Holguín et al., (2014) se muestra cómo con el uso de estos sistemas se logra ordenar frutas tropicales latinoamericanas en base a características dadas, también se destaca la labor de De La Cruz & Donoso, (2016) quienes lograron desarrollar una máquina que permite clasificar objetos prismáticos triangulares, rectangulares, cilindros de colores rojo, verde y azul para un laboratorio de automatización industrial. El documento de Peršak et al., (2020), expone que los sistemas de clasificación y visión artificial pueden ser usados para identificar de manera eficiente plástico granulado de policarbonato transparente, así también, los aplicativos de estos sistemas se extiende al control de un agarre robótico para la clasificación automática de plástico entre un bulto de basura (Zhihong et al., 2017). El uso de brazos robóticos con visión artificial como clasificadores extiende su utilidad al permitir ordenar objetos según su forma (Moncada, 2018).

Por otro lado, existen servicios de computación en la nube enfocados en la creación, prueba y despliegue de aplicaciones mediante el uso de centro de datos, tales como Azure de Microsoft, AWS de Amazon, servicios de Google y más, cada uno con herramientas de cómputo que mediante su uso facilitan el desarrollo de proyectos. Dentro de estos servicios web encontramos APIs que cuentan con herramientas de visión por computador e inteligencia artificial. Siendo de interés el desarrollo de un sistema de clasificación mediante el análisis de imágenes y el consumo de estos servicios. El manejo de estas herramientas brinda facilidad para el control del hardware, pues consigue que el procesamiento del algoritmo principal no se lo realice de manera local. Hasta ahora los trabajos revisados buscaron como objetivo la clasificación de una categoría especifica, ya sea, frutas, plástico, huevos, flores etc. Es por ello por lo que otro punto a destacar en la idea a proponer en este trabajo es la capacidad que poseerá el sistema de generar entrenamiento con artículos variados para así poder clasificar

diversas categorías y así múltiples objetos empleando una misma máquina, y todo esto con la ayuda de un servicio web centrado en visión artificial.

Justificación

La implementación de sistemas que incorporan visión por computador e inteligencia artificial constituyen una solución atractiva para industrias que cuentan con procesos de inspección de calidad o clasificación de objetos, debido a que permiten automatizar tareas manuales, reduciendo riesgos por error humano, considerando que las decisiones tomadas por un operario se ven afectadas por factores psicológicos como puede ser la fatiga, estrés u otros, más aún cuando la tarea a realizar es repetitiva, de apreciación o percepción. En este contexto el estudio realizado por Paulus et al., (1997), en el que personal fue sometido a pruebas para comparación de parámetros tales como forma, tamaño y color de varias manzanas, el resultado fue la incapacidad de una correcta selección de la fruta bajo los requerimientos solicitados. Es así como características de los sistemas de clasificación como, la objetividad y consistencia en largos periodos de tiempo, se convierten en ventajas dentro una línea de producción.

Bajo el estudio de trabajos realizados de manera previa, ha quedado demostrado que un sistema de clasificación puede ser usado para tareas como las de separar tipos de café, seleccionar huevos en buen estado, detectar tipos de flores, y más, mejorando así la calidad del producto y aumentando el volumen de producción. Si bien la implementación de sistemas de automatización permite mejoras en las líneas de producción, estas serán aún mayores si se añaden el uso de herramientas como lo son los servicios web, los cuales son eficaces por su capacidad de conexión con la nube y el almacenamiento o descarga de información desde cualquier lugar, por lo que, esta herramienta da una versatilidad a un sistema clasificador. El deseo de generar un sistema de clasificación configurable que permita cambiar el tipo de objeto a clasificar empleando una misma máquina, puede llegar a requerir características altas de

hardware, siendo una alternativa para reducir este requerimiento el uso de servicios con procesamiento en la nube.

Importancia

El desarrollo del presente proyecto busca no solo ser un aporte tecnológico, sino también ser parte de un campo de investigación que abarque el uso de herramientas tales como la visión por computador e inteligencia artificial y su vinculación con el consumo de servicios web, y estas permitan generar un cambio en la idea tradicional de matriz productiva en las industrias, obteniendo los beneficios operativos de seguridad, flexibilidad, reducción en el tiempo de ciclo productivo, precisión al eliminar factores subjetivos en la inspección, mejora de calidad, ahorro de recursos, etc. Sumado a las ventajas antes mencionadas se presenta las características favorables del uso de servicios web, pues existe eficiencia y ahorro al usar una infraestructura en la nube, se evita el procesamiento local y el requerimiento de equipos, otro aspecto a destacar es la seguridad en el manejo de datos que se posee al hacer uso de dichos servicios. Finalmente dar al sistema la capacidad de una supervisión de datos del proceso de manera remota será una característica de valor dentro del proyecto. El concluir de manera exitosa con la propuesta de este trabajo, muestra que la integración de estas herramientas tecnológicas dentro de procesos de clasificación realizados en la actualidad, en su mayoría de manera manual, es posible, y sobre todo beneficioso.

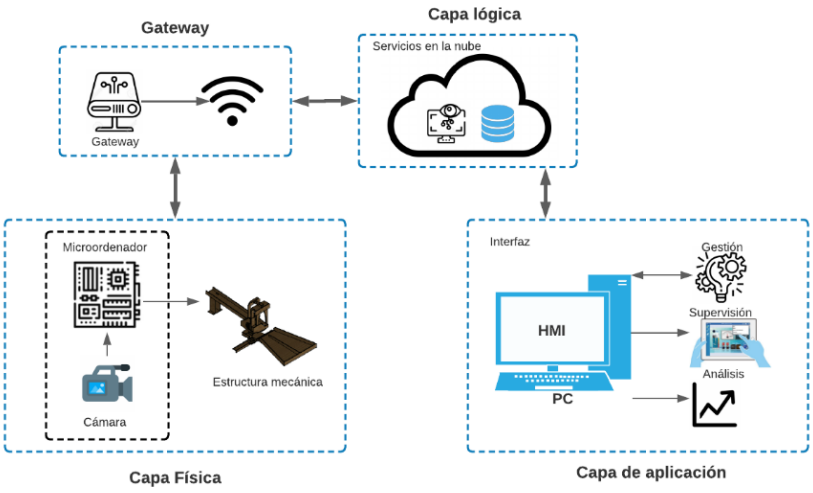
Alcance

Se plantea la implementación de un prototipo de un sistema clasificador de objetos basado en reconocimiento de imágenes y servicios en la nube. El usuario podrá seleccionar de una lista de opciones presentadas en una interfaz web los objetos que en ese momento el sistema podrá clasificar, de no existir el objeto dentro de la lista predeterminada, se podrá acceder a una opción dentro de la interfaz que permita el ingreso de imágenes del nuevo elemento, para el reentrenamiento del modelo de clasificación. Una vez seleccionados los

objetos el usuario los ingresará de manera individual, y el hardware del sistema permitirá la clasificación de estos en dos grupos en base a características predefinidas (forma, color) hasta una nueva selección. Para ello se ilustra en la figura 1 el esquema del sistema, mismo que cuenta con:

Figura 1

Esquema del sistema.



A nivel físico, este contendrá el sistema de control y mando. Está compuesto por una cámara como equipo para adquisición de datos (imágenes), un microordenador (Raspberry Pi) para el envío de los datos adquiridos hacia la nube, así también este será el encargado de comandar los actuadores de la estructura mecánica. Se considera que el microordenador deberá poseer características que permitan las acciones antes mencionadas de adquisición de datos, transmisión de datos y generación de señales de control. Por otro lado, el sistema mecánico poseerá una estructura, tal que, permita la clasificación de objetos en dos grupos durante cada proceso de ejecución, para ello contará con actuadores electromecánicos que faciliten esta función.

Para que exista comunicación bidireccional y así la transmisión de datos sea posible entre la capa física y la nube, será necesario del uso de un dispositivo como puerta de enlace (Gateway). Para reducir la latencia producida por la comunicación entre el ordenador y la nube se optará por una conexión ethernet estimando una reducción de latencia. A nivel lógico se encuentra el algoritmo central del sistema clasificador, mismo que debido a su característica de poder ser reentrenado para cada nuevo objeto ingresado, requerirá de propiedades de procesamiento y almacenamiento de datos altas, para lo cual se plantea el uso de herramientas basadas en el consumo de servicios en la nube (Azure), para este caso específico, se hará uso de servicios cognitivos de visión artificial (Computer Vision), basado en el uso de redes neuronales profundas, adaptándolos para la generación de un modelo de clasificación de imágenes que sea útil para cada caso. Por cada ciclo de clasificación se generará un registro de datos, mismos que serán almacenados en una base en la nube. El modelo para generar debe poseer un equilibrio entre su rendimiento, eficacia y la latencia de clasificación, para este fin se propone variar el tamaño del modelo haciéndolo más pequeño mediante la reducción de la cantidad de atributos de entrada o la complejidad del modelo. Para este prototipo se desea tener como mínimo una clasificación de 8 objetos por minuto. Finalmente, todo el sistema será controlado y supervisado mediante una interfaz humano-máquina (HMI) en un PC, la cual tendrá herramientas de control, para la selección del tipo de objeto a clasificar por el sistema, herramientas de supervisión, para visualizar el estado del sistema en funcionamiento o en parada, y herramientas de análisis, permitiendo la adquisición de datos generados por el sistema clasificador, además contará con una interfaz de supervisión web/móvil, para visualizar el estado de la máquina y datos relevantes del proceso, El sistema de clasificación propuesto requiere el diseño de un modelo de clasificación para múltiples objetos, el dominio de servicios web y en específico de los servicios de inteligencia artificial y servicios cognitivos, la creación de un software, mostrado mediante una interfaz web para el uso del modelo de clasificación,

así como para presentar datos del sistema, la generación de la red para su interconexión con la nube, finalmente se deberá diseñar un modelo mecánico y electrónico para su estructura a nivel físico.

Objetivos

General

Implementar un prototipo de un sistema de clasificación binaria basado en el uso de servicios web cognitivos y redes neuronales profundas

Específicos

- Identificar fuentes de información relevantes para el desarrollo del estado del arte para sistemas de clasificación con imágenes.
- Determinar que metodologías se usarán para la aplicación de los algoritmos de identificación y clasificación basados en el consumo de servicios web cognitivos.
- Ajustar un modelo para su uso en el sistema de clasificación de múltiples objetos.
- Diseñar una interfaz que permita el uso del modelo de clasificación.
- Diseñar un prototipo mecánico y electrónico, tal que, su estructura permita la clasificación de objetos en dos grupos determinados.
- Generar una arquitectura de integración de los componentes mecánicos y electrónicos con los servicios en la nube para su gestión mediante una interfaz web/móvil.
- Evaluar el sistema mediante pruebas de carga, usabilidad.
- Publicar en la plataforma de desarrollo colaborativo de software GitHub el proyecto y sus resultados.

Estado del Arte

Una vez planteado el alcance y objetivos del proyecto es necesario abordar el estado del arte como modalidad de investigación documental, y así, poder adquirir el conocimiento dentro del área de estudio de interés, en este caso, sobre el uso de sistemas de visión artificial, redes neuronales, y sistemas de clasificación. Para ello se plantea el desarrollo de un Mapeo Sistemático de la Literatura (SMS, Systematic Mapping Study) sobre sistemas de clasificación basadas en visión artificial y uso de servicios web cognitivos.

Mapeo Sistemático de la Literatura

Basado en la metodología desarrollada por Petersen et al.,(2015) para la realización de un SMS, la cual estructura un área de investigación, se plantean los siguientes pasos:

Preguntas de investigación. En la tabla 1 se plantean las preguntas que permiten indagar sobre cómo se han desarrollado procesos relacionados al estudio y uso de visión artificial para sistemas de clasificación basados en el uso de servicios web.

Tabla 1

Mapeo sistemático.

Preguntas de investigación	Motivación
RQ1: ¿Cuáles son las investigaciones desarrolladas alrededor de un sistema clasificador basado en visión artificial?	Conocer investigaciones que aborden el uso de imágenes para la clasificación de objetos.
RQ2: ¿Qué técnicas y/o tecnologías existen para el desarrollo de software para su uso en sistemas de clasificación y visión artificial?	Conocer técnicas y tecnologías que se han usado para desarrollar software como herramienta para sistemas de clasificación.

Preguntas de investigación	Motivación
RQ3: ¿Qué tipo de servicios web existen para su uso en procesos de clasificación de imágenes?	Conocer los servicios web cognitivo disponibles para su uso en la clasificación de imágenes.
RQ4: ¿Qué tipo servicios web cognitivos han sido adaptados para su uso dentro de un sistema de clasificación?	Conocer que servicios web cognitivos ya han sido aparatados en procesos de clasificación de objetos.

Criterios de búsqueda primarios y secundarios. Para la búsqueda y recopilación de investigaciones, se plantea el uso de librerías digitales tales como Springer, IEE Xplore, MDPI, además de Google académico.

Debido a que se trata de un primer reconocimiento al estado de arte, y se busca conocer la tecnología existente y que ha sido empleada para el tema del proyecto, no se restringe el uso de información proveniente de ninguna publicación como tesis, artículos, páginas web, etc.

Definir Cadena de búsqueda. La tabla 2 contiene los términos definimos como relevantes para poder generar las cadenas de búsqueda en las librerías digitales.

Tabla 2

Cadenas de búsqueda.

Preguntas de investigación	Términos alternativos
[RQ1]: Sistema clasificador	system for classification OR sorters
[RQ1]: Visión artificial	Artificial vision OR Computer vision
[RQ2]: Software	Software OR Computer program
[RQ3]: Clasificación de imágenes	Image classification OR Object classifier

Preguntas de investigación	Términos alternativos
[RQ4]: Servicios web cognitivos	Web services OR Cloud computing

La cadena de búsqueda general para el SMS queda como:

(System for classification OR “sorters”) AND (“Artificial vision” OR Computer vision) AND (Software OR Computer program) AND (Image classification OR Object classifier) AND (web services OR Cloud computing)

Con esta cadena general se obtienen un total de 15 resultados de búsqueda, por lo que se establecen combinaciones de términos que permitan generar nuevas cadenas de búsquedas, así se obtienen las siguientes:

Cadena de búsqueda 1:

(System for classification OR “sorters”) AND (“Artificial vision” OR Computer vision) AND (Image classification OR Object classifier) AND (web services OR Cloud computing)

Cadena de búsqueda 2:

(System for classification OR "sorters") AND (“Artificial vision” OR Computer vision) AND (Software OR Computer program)

Cadena de búsqueda 3:

(System for classification OR "sorters") AND (“Artificial vision” OR Computer vision) AND (web services OR Cloud computing)

Para la combinación de cadenas se obtiene un número considerable de respuestas por ello se continua con el siguiente paso del SMS.

Fijar criterios de inclusión y exclusión. Se estable como criterios de inclusión y exclusión los siguientes apartados:

- Que los documentos sean de acceso libre.
- Artículos completos.

- El año de publicación sea desde 2018 hasta el presente año.
- Que pertenezca a la categoría de ingeniería eléctrica y electrónica, sistemas de computación, ciencias de la computación e inteligencia artificial.
- No se excluyó los tipos de documentos, sea tesis doctorar, artículos de revista, etc.
- No se excluye documentos por el autor.

Determinar la estrategia de extracción de datos. Para la extracción y síntesis de información se elige como herramienta una ficha de extracción de información, en la que se registran datos bibliográficos, así como hallazgos del artículo o publicación.

Clasificación de los artículos. El proceso de extracción de datos se realiza en tres fases detalladas a continuación:

1. La primera fase considera **Title + Abstract + Keywords**, obteniendo un total de 3429 resultados, entre todas las bibliotecas digitales.
    - IEEE Xplore: 1485
    - Springer Link: 1935
    - MDPI: 9
2. Para la segunda fase se consideran los criterios de inclusión y exclusión antes detallados, y atendiendo a **Title + Abstract + Keywords,** obteniendo un total de 313 resultados, entre todas las bibliotecas digitales.
    - IEEE Xplore: 126
    - Springer Link: 178
    - MDPI: 9
3. La tercera y última fase se aplicó un filtro mucho más exhaustivo en que se consideran solo los artículos de interés y de aporte. Teniendo en cuenta **Title +**

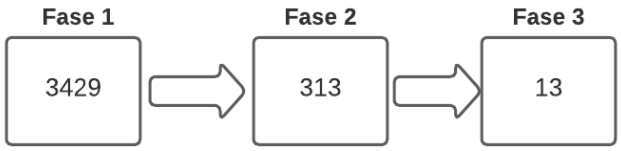
Abstract + Keywords + Full Article, obteniendo un total de 13 resultados, entre todas las bibliotecas digitales.

- IEEE Xplore: 6
- Springer Link: 6
- MDPI: 1

Resultados. La figura 2 ilustra que una vez realizado los pasos correspondientes para el SMS se obtiene un resultado de 13 trabajos de interés dentro del área de investigación.

Figura 2

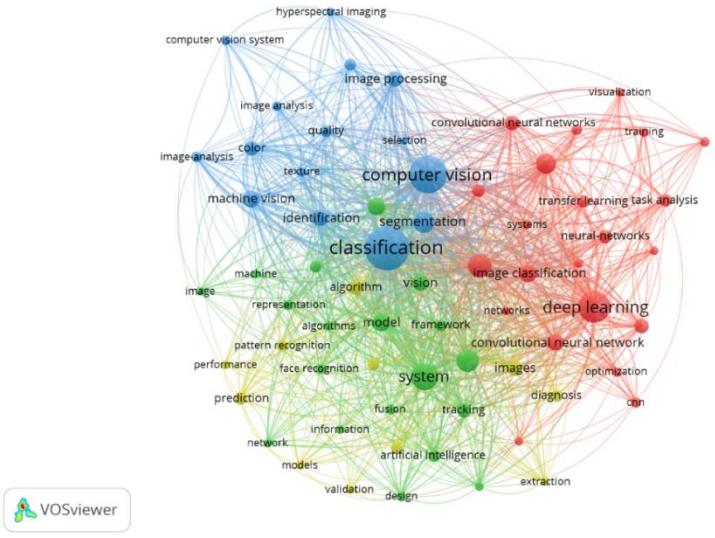
Cantidad total de artículos en cada fase del SMS.



A continuación, se detallan datos significativos correspondientes a las investigaciones existentes en el área de investigación de interés. Se empleó el software VOSviewer para la elaboración de mapas de densidad, en él se introdujeron las siguientes palabras claves: “classification”, “Computer visión” y “Cloud computing”. La figura 3 muestra que la palabra con más búsqueda en librerías digitales de investigación es “classification” seguido de “computer visión”.

Figura 3

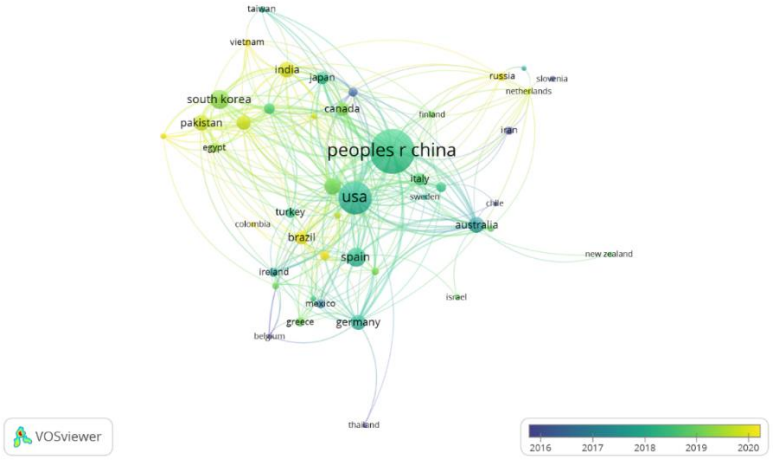
Densidad de búsqueda por palabras.



En la figura 4 se puede observar el mapa de densidad según los aportes de investigación dado por cada país, dando como resultado que el país que más aportes tiene dentro del tema es China, seguido por Estados Unidos y Corea del Sur.

Figura 4

Densidad de búsqueda por países.



Conclusiones. La elaboración del SMS da como resultado la respuesta a las preguntas planteadas anteriormente, fundamentadas y sustentadas por artículos o trabajos de investigación que están dentro del área de interés en el desarrollo del proyecto.

RQ1: ¿Cuáles son las investigaciones desarrolladas alrededor de un sistema clasificador basado en visión artificial?

Como se ha podido observar, existen diversos trabajos haciendo uso de la visión artificial en un sistema clasificador, lo cual indica que estos tipos de sistemas son versátiles y muy usados en los procesos de producción, se pueden realizar clasificadores inteligentes de frutas con el uso de redes neuronales (Gnana et al., 2019), de esta manera mejora la calidad del producto, el desarrollo de los sistemas clasificadores van avanzando cada vez más, en la actualidad ya no se clasifican simplemente con características muy distintivas, se ha logrado poder diferencias pequeños rasgos entre los distintos objetos, como saber el nivel de madurez, el peso y la fecha de cosecha de la fruta (Faisal et al., 2020), o también con la ayuda de controladores se puede mejorar el sistema mediante un controlador Fuzzy en el cual su objetivo principal es diferenciar los colores o alguna característica relevante (Abad et al., 2018).

RQ2: ¿Qué técnicas y/o tecnologías existen para el desarrollo de software para su uso en sistemas de clasificación y visión artificial?

Existe un crecimiento de diversas técnicas las cuales implican el desarrollo de clasificadores basados en visión artificial, como base fundamental para estos proyectos se usa diversos softwares que permitan la flexibilidad de unir diferentes tecnologías, como el uso del lenguaje Arduino y creación de una interfaz para la clasificación de huevos (Quilloy et al., 2018), con la aparición de los softwares para la implementación de visión artificial se ha ido creando comunidades para dar soporte a diferentes lenguajes de programación, o creación de diversas librerías para facilitar el uso de estas tecnologías, la mayoría de estas tecnologías son

de código abierto, por lo cual se puede hacer uso de estos y realizar diversos sistemas clasificadores (Amaya Zapata et al., 2016), debido a esta facilidad también se crean librerías con diversos módulos para inteligencia artificial que son de fácil uso y tienen una confiabilidad muy elevada (Zhao et al., 2022), por lo que, son muy usadas en el desarrollo de un software.

RQ3: ¿Qué tipo de servicios web existen para su uso en procesos de clasificación de imágenes?

Los servicios en la nube son un servicio de software que proporciona soluciones comerciales consumidas por diferentes solicitantes de servicios y al que se puede acceder mediante un protocolo web estándar (Subbulakshmi et al., 2020). Iniciamos mencionando al servicio de cloud computing como aquel servicio en la nube que permite el acceso remoto a software, almacenamiento y procesamiento de datos. El procesamiento en la nube, el big data y tecnología de aprendizaje profundo, como ejemplo de su funcionalidad, son usados para análisis de resultados médicos (Lie et al., 2020). Otro tipo de servicios web son aquellos denominados servicios cognitivos, mismos que dotan a la máquina con capacidades de percibir y procesar esos datos, por medio de visión, voz, conocimiento, lenguaje, semejantes a la de los humanos. El uso de servicios cognitivos de visión permite analizar videos provistos por cámaras instaladas en áreas urbanas para la detección de basura y generar alerta a la municipalidad (Coccoli et al., 2022). Otro tipo de servicio se da en la ejecución y canalización para una optimización evolutiva en Azure Batch Service, para la administración de un conjunto de nodos del proceso, en lugar de usar máquinas virtuales (Wierzbiński et al., 2021). Estos son ejemplos del tipo de servicios web disponibles y usados para proyectos tecnológicos.

RQ4: ¿Qué tipo servicios web cognitivos han sido adaptados para su uso dentro de un sistema de clasificación?

Muchas empresas utilizan modelos de clasificación creados por proveedores externos, como Google Cloud Visión, Amazon Rekognition, Computer Vision de Microsoft Azure .(Vermeire et al., 2022). La API Face de Azure, es empleada para la detección de rostros en HoloLens, mediante una llamada remota al servicio, en la que se analiza su velocidad de respuesta y la precisión de detección (Huang et al., 2019). Existen también servicios disponibles para la creación de modelos de visión artificial por medio de la carga de imágenes, tal como Microsoft Azure Custom Vision, la cual se basa en una CNN preentrenada y ofrece una técnica de transferencia de aprendizaje para los usuarios, simplemente importando las imágenes de entrenamiento y marcándolas en un generador de modelos (Gnana et al., 2019).

Otras conclusiones

Una vez solventadas las preguntas de investigación, es oportuno agregar conclusiones que pueden ayudar al desarrollo del trabajo:

- Los sistemas clasificadores ya no son únicamente un aparato mecánico, se han ido adaptando a las necesidades de los distintos campos de producción, por lo que, en la actualidad estos sistemas son potenciados con diversos lenguajes de programación, por medio de los cuales se hace uso de la inteligencia artificial, gracias a esta característica se puede mejorar los distintos procesos en los cuales son necesarios clasificar imágenes basados en características especificadas por el usuario.
- Los servicios en la nube tienen como principal componente al Cloud computing, y en base a ello el desarrollo de servicios tales como Custom Vision, Amazon Rekognition y más.

- Los servicios en la nube se extienden también al almacenamiento y análisis de datos, abarcando bases de datos e incluso involucrando al big data y su tratamiento de estos.
- Los trabajos de investigación y desarrollo tecnológico relacionados al tratamiento de imágenes y uso de servicios en la nube muestran la factibilidad técnica del proyecto bajo la guía de los trabajos antes mencionados, sin embargo, deberá extenderse para el propósito de clasificación de objetos de manera general.

Revisión Sistemática de la Literatura

Una vez culminado el SMS, surge la necesidad de recopilar y sintetizar evidencias y hallazgos dentro del área de investigación para ellos se emplea el método SLR que apunta a sintetizar la evidencia, considerando también la solidez de la misma, que a diferencia de los mapas sistemáticos, los cuales preocupan principalmente estructurar un área de investigación (Petersen et al., 2015). Sin embargo, los pasos para su desarrollo son similares.

Preguntas de investigación. En la tabla 3 se plantean las preguntas de investigación.

Tabla 3

Preguntas de investigación.

Preguntas de investigación	Motivación
¿Qué tecnologías existen para el desarrollo de un sistema clasificador basado en visión artificial?	
RQ1: ¿Qué técnicas y/o tecnologías permiten la clasificación mediante el reconocimiento de imágenes?	Conocer los diversos algoritmos y el nivel de madurez de estos dentro de sistemas de clasificación de imágenes.
RQ2: ¿Qué técnicas de desarrollo de software basadas en redes neuronales	Determinar lenguajes de programación, protocolos y otras características de software

Preguntas de investigación	Motivación
permiten la elaboración de herramientas para su uso en sistemas de clasificación?	necesarias para el desarrollo y uso de los algoritmos usados en un sistema de clasificación.
¿Qué tipo de servicios web existen para su uso en procesos de clasificación de imágenes?	
RQ3: ¿Qué servicios web o servicios en la nube brindan la capacidad de clasificación mediante el reconocimiento de imágenes?	Conocer los diferentes servicios web disponibles, determinar sus características y rendimiento en procesos de clasificación de imágenes.
RQ4: ¿Qué técnicas de desarrollo de software permiten el uso de servicios web para el procesamiento en la nube y clasificación de imágenes?	Determinar lenguajes de programación, protocolos y otras características de software necesarias para el manejo de servicios web.

Criterios de búsqueda primarios y secundarios. Para la búsqueda y recopilación de investigaciones, los criterios serán similares a los expuestos en el SMS, se mantiene el uso de librerías digitales como Springer, IEE Xplore, MDPI, y Google académico.

Debido a que se busca mayor profundidad en técnicas y tecnologías, se excluyen publicaciones con un punto de vista meramente de divulgación, tales como libros, páginas web, artículos de divulgación, data papers, etc., dando un enfoque a artículos de investigación.

Definir Cadena de búsqueda. La tabla 4 contiene los términos definimos como relevantes para poder generar las cadenas de búsqueda en las librerías digitales, estos son dependientes de las preguntas de investigación para el SLR antes mencionadas.

Tabla 4

Cadenas de investigación.

Preguntas de investigación	Términos alternativos
[RQ1]: Sistema de clasificación	System for classification OR sorters
[RQ1]: Reconocimiento de imágenes	Image recognition OR Computer vision
[RQ2]: Redes neuronales	Neural networks OR networks
[RQ2]: Desarrollo de software	software development OR program
[RQ3]: Servicios en la nube	Web services OR Cloud services
[RQ4]: Procesamiento en la nube	Cloud computing OR network computing

La cadena de búsqueda general para el SMS queda como:

(System for classification OR “sorters”) AND (Image recognition OR Computer vision) AND (“Neural networks” OR networks) AND (Software development OR program) AND (Web services OR cloud services) AND (“Cloud computing” OR network computing)

Con esta cadena general se obtienen un total de 24 resultados de búsqueda, por lo que se establecen combinaciones de términos que permitan generar nuevas cadenas de búsquedas así se obtienen las siguientes:

Cadena de búsqueda 1:

(System for classification OR “sorters”) AND (“Neural networks” OR networks) AND (Web services OR cloud services) AND (“Cloud computing” OR network computing)

Cadena de búsqueda 2:

(System for classification OR “sorters”) AND (Image recognition OR Computer vision) AND (“Neural networks” OR networks) AND (Software development OR program) AND (Web services OR cloud services)

Cadena de búsqueda 3:

(System for classification OR “sorters”) AND (“Neural networks” OR networks) AND (Software development OR program) AND (Web services OR cloud services) AND (“Cloud computing” OR network computing)

Para la combinación de cadenas se obtiene un número considerable de respuestas por ello se continua con el siguiente paso del SLR.

Fijar criterios de inclusión y exclusión. Se estable como criterios de inclusión y exclusión los siguientes apartados:

- Que los documentos sean de acceso libre.
- Artículos completos.
- Trabajos realizados en revistas o conferencias.
- El año de publicación sea desde 2018 hasta el presente año.
- Que pertenezca a la categoría de: Ingeniería eléctrica y electrónica, sistemas de computación, ciencias de la computación e inteligencia artificial.
- Se excluyó los tipos de documentos de divulgación como libros, artículos de divulgación, data papers, reviews, etc.
- No se excluye documentos por el autor.

Determinar la estrategia de extracción de datos. Para la extracción y síntesis de información se continua con el uso de una ficha de extracción de información.

Clasificación de los artículos. El proceso de extracción de datos se realiza en tres fases detalladas a continuación:

1. La primera fase considera **Title + Abstract + Keywords**, obteniendo un total de 3873 resultados, entre todas las bibliotecas digitales.
    - **IEEE Xplore:** 652
    - **Springer Link:** 3218

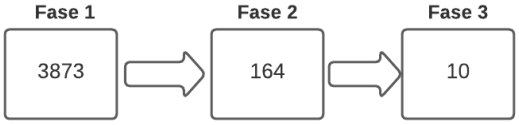
- **MDPI:** 3

2. Para la segunda fase se consideran los criterios de inclusión y exclusión antes detallados, y atendiendo a **Title + Abstract + Keywords,** obteniendo un total de 164 resultados, entre todas las bibliotecas digitales.
    - **IEEE Xplore: 42**
    - **Springer Link: 119**
    - **MDPI: 3**
3. La tercera y última fase se aplicó un filtro mucho más exhaustivo en que se consideran solo los artículos de interés y de aporte. Teniendo en cuenta **Title + Abstract + Keywords + Full Article,** obteniendo un total de 10 resultados, entre todas las bibliotecas digitales.
    - **IEEE Xplore: 5**
    - **Springer Link: 4**
    - **MDPI: 1**

Resultados. La figura 5 ilustra que una vez realizado los pasos correspondientes para el SLR se obtiene un resultado de 10 trabajos de interés dentro del área de investigación.

Figura 5

Cantidad total de artículos en cada fase del SMS.

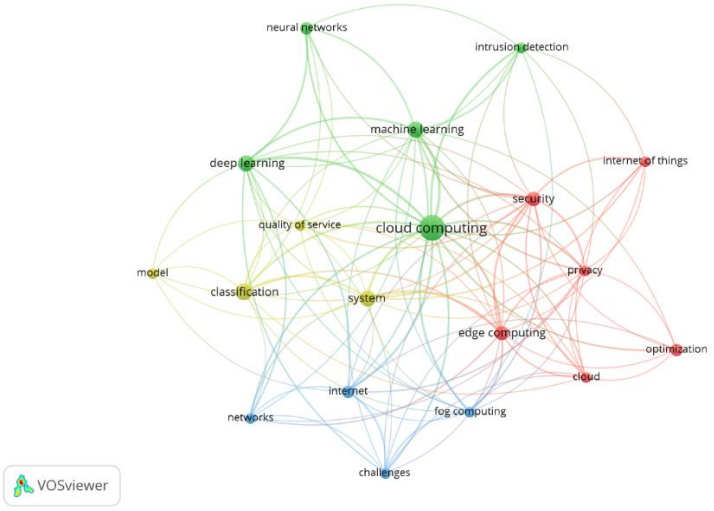


A continuación, se detallan datos significativos correspondientes a las investigaciones existentes en el área de investigación de interés, se introdujeron las siguientes palabras claves: “Cloud computing”, “web services”, y “neural networks”. La figura 6 muestra que la palabra con

más búsqueda en librerías digitales de investigación es “Cloud computing” seguido de “machine learning”.

Figura 6

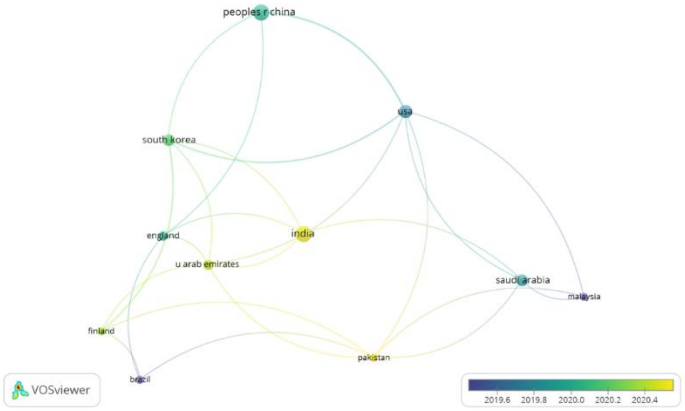
Densidad de búsqueda por palabras.



En la figura 7 se puede observar el mapa de densidad según los aportes de investigación dado por cada país, dando como resultado que el país que más aportes tiene dentro del tema es China, seguido por la India y Estados Unidos.

Figura 7

Densidad de búsqueda por países.



Conclusiones. La elaboración del SLR da como resultado la respuesta a las preguntas planteadas anteriormente.

RQ1: ¿Qué técnicas y/o tecnologías permiten la clasificación mediante el reconocimiento de imágenes?

Los principales trabajos sobre inteligencia artificial, en el campo de reconocimiento de imágenes se han desarrollado usando redes neuronales convolucionales (CNN), esto indica que el uso más frecuente y recomendable para el reconocimiento de algo en particular mediante imágenes es mediante CNN, sin importar el ámbito de estudio como lo puede ser en la medicina (Wang, 2020), el uso en este campo ayuda a prevenir que las enfermedades agraven, sin embargo, el manejo de esta tecnología es empleada comúnmente en industrias para mejorar su productividad, poder clasificar de mejor manera sus productos (Li et al., 2019), también es importante la calidad de imagen que se ingresa al algoritmo para que la red pueda predecir, por lo que, también existen trabajos que pueden clasificar texturas de imágenes para el uso adecuado de la CNN (Song et al., 2016).

RQ2: ¿Qué técnicas de desarrollo de software basadas en redes neuronales permiten la elaboración de herramientas para su uso en sistemas de clasificación?

Entre los lenguajes usados para el desarrollo de software tenemos Python, M de Matlab, C++, C# y java, junto con diversas librerías que permiten potenciar el uso de estos, sin embargo, varios artículos muestran el potencial aumenta al unir diferentes lenguajes de programación (Kolivand et al., 2021), (Bernacki, 2021), esto permite obtener las mejores características de cada lenguaje y de esta manera se desarrollan softwares con mejor precisión a la hora de realizar diversas predicciones, no obstante, no es necesario mezclar varios

lenguajes para poder realizar una clasificación con el uso de redes neuronales, se pude utilizar simplemente Python para el desarrollo de inteligencia artificial (Tang et al., 2020).

RQ3: ¿Qué servicios web o servicios en la nube brindan la capacidad de clasificación mediante el reconocimiento de imágenes?

Hay diferentes formas de implementar una solución de reconocimiento de objetos. Un enfoque común es ejecutar el proceso de reconocimiento de forma remota en servicios en la nube como Google Cloud Vision, Microsoft Azure Computer Vision, Amazon Rekognition, etc. Estos servicios ya están entrenados con grandes conjuntos de datos que permiten un mejor rendimiento (Valipoor & de Antonio, 2022). El servicio de Google Cloud Vision fue adaptado de tal forma que se desarrolló un sistema móvil para reconocer objetos, textos y rostros (Bharatia et al., 2019)

RQ4: ¿Qué técnicas de desarrollo de software permiten el uso de servicios web para el procesamiento en la nube y clasificación de imágenes?

Las funciones como servicio (FaaS), son un tipo de servicio de cloud computing que dan la posibilidad de diseño, ejecución y gestión de aplicaciones, estas funciones se centran en la escucha de eventos para la ejecución de funciones individuales (Salmerón, 2020). Para uso de estas funciones se sabe que Azure Functions admite varios lenguajes de programación de uso común (C#, JavaScript, F#, Java, PowerShell, Python, Typescript). La plataforma FaaS que ofrece Google, de manera similar a AWS Lambda y Azure Functions, admite lenguajes de programación (Node.js, Python, Go y Java). Amazon, Microsoft y Google Cloud Functions usan disparadores para ejecutar funciones (Bocci et al., 2021).

Otras conclusiones

Una vez contestadas las preguntas de investigación, es importante agregar conclusiones que pueden ayudar al desarrollo del trabajo:

- Al igual que sucedió en el SMS, se ha podido encontrar diversos trabajos que desarrollan investigación basados en el uso de clasificadores con inteligencia artificial, y con el SLR se logra completar la información que se requiere para el desarrollo correcto del trabajo.
- La creación de diversos modelos para la clasificación de imágenes es basada en CNN en su mayoría, por lo que ha demostrado tener una gran efectividad al momento de predecir.
- El costo del consumo de cada servicio según el proveedor será un factor de decisión, debido a que la funcionalidad y precisión de cada uno es similar y de alto rendimiento en lo que a análisis de imágenes se refiere.
- El manejo de varios lenguajes de programación para el uso de las funciones de Azure, Amazon y Google, facilitan al desarrollador su consumo.
- Los servicios en la nube, enfocados en el tratamiento de imágenes brindan un abanico amplio de funcionalidades, como reconocimiento de rostros, análisis de video, análisis de texto o identificación de objetos. Por ello se convierten en una herramienta de gran utilidad que deberá ser adaptada para su uso con un objetivo específico.

Capítulo 2: Marco Conceptual

Sistemas de clasificación

Se denomina sistema de clasificación a aquel conjunto de elementos que, relacionados entre sí, permiten el ingreso de un objeto, el cual será asignado y etiquetado dentro de una categoría especifica. El sistema basa su funcionamiento sobre un conjunto de reglas definidas, con base en cierta información extraída del objeto, como características de color, forma, tamaño, etc. De manera general un sistema de clasificación o sorter, consta de una estructura mecánica y un algoritmo que lo comanda.

Tipos de clasificadores

Dentro del campo de la industria los clasificadores son una herramienta para la producción, encargadas del transporte y clasificación del producto, debido a la gran cantidad de objetos que son necesarios clasificar, los tipos de estructuras mecánicas que conforman esta herramienta, son igual de extensos, en GEICOM (2020) se menciona algunos de ellos:

- Clasificador desviador: está compuesto por un transportador tipo banda, y un mecanismo electroneumático o eléctrico que se acciona empujando el producto hacia la salida deseada.
- Clasificador Pop up: desvía el producto mediante un mecanismo ubicado en cada salida.
- Clasificador de bandas cruzadas: empleado mucho por industrias de envíos, se caracteriza por la unión de ramales hacia la ruta de transporte y llegada.
- Clasificador vertical: facilita la clasificación de productos en niveles superior media o bajo.

Inteligencia artificial

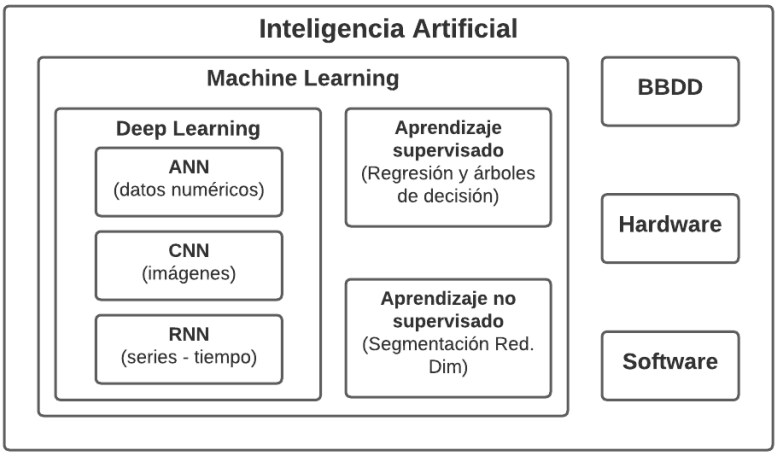
Se define a la inteligencia artificial (IA) como un campo dentro de la computación el cual busca principalmente la transmisión de inteligencia antropomórfica y pensamiento en

máquinas, dotando a la maquinaria de capacidades de aprendizaje autónomo, adaptación a circunstancias específicas y autocorrección de sus errores, es decir, que pueda pensar por sí misma (Ting et al., 2021). La inteligencia artificial abarca subcampos relacionados entre sí como se ilustra en la figura 8 y se menciona los siguientes:

- Machine Learning (ML): también llamado aprendizaje automático, tiene como principio central que las máquinas tomen datos y aprendan por sí mismas. ML, Arthur Samuel pionero en su desarrollo, lo definió como un "campo de estudio que brinda a las computadoras la capacidad de aprender sin ser programadas explícitamente". ML se centra principalmente en la clasificación y la regresión en función de características conocidas, aprendidas previamente de los datos de entrenamiento (Xin et al., 2018). Dentro de este campo existe el aprendizaje supervisado, no supervisado y Deep Learning.
- Deep Learning (DL): el aprendizaje profundo es un subconjunto de ML. Utiliza algunas técnicas de ML para resolver problemas del mundo real aprovechando las redes neuronales que simulan la toma de decisiones humana (Hope, 2017). El aprendizaje profundo o red neuronal profunda se refiere a las redes neuronales artificiales (ANN) con múltiples capas, caracterizada por manejar una gran cantidad de datos, y por superar el desempeño de los métodos clásicos, especialmente en el reconocimiento de patrones.
- Bases de Datos (BBDD), Hardware, Software: al hablar de bases de datos e inteligencia artificial, se hace referencia al Big Data, su almacenamiento y trato, sobre el hardware se toma en consideración el cómputo distribuido o al cómputo en la nube para el procesamiento de algoritmos, al hablar de software engloba herramientas tales como R, Python, etc., que facilitan la generación y uso de modelos de IA.

Figura 8

División de la inteligencia artificial.



Redes neuronales

Se denomina redes neuronales a modelos matemáticos que utilizan algoritmos de aprendizaje inspirados en el cerebro para almacenar información (Brown et al., 2018). El término redes neuronales abarca una familia de métodos computacionales no lineales, que pretenden imitar el funcionamiento del cerebro humano (Marini, 2009).

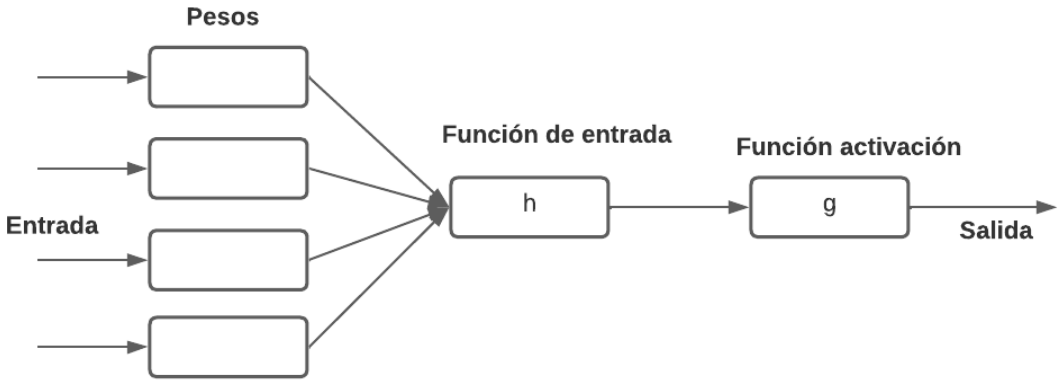
Redes neuronales artificiales (ANN).

Las redes neuronales artificiales reúnen su conocimiento al detectar patrones y relaciones en los datos y aprenden a través de la experiencia, no de la programación. Una ANN está formada por cientos de unidades individuales, neuronas artificiales o elementos de procesamiento (PE), conectados con coeficientes (pesos), que constituyen la estructura neuronal y están organizados en capas. La manera de comportarse de una red neuronal está determinada por las funciones de transferencia de sus neuronas, por la regla de aprendizaje y por la propia arquitectura. Los pesos son los parámetros que son ajustables, convirtiendo una red neuronal en un sistema parametrizado. La suma ponderada de las entradas constituye la

activación de la neurona. La señal de activación pasa a través de la función de transferencia para producir una salida única de la neurona. La función de transferencia introduce no linealidad en la red (Agatonovic-Kustrin & Beresford, 2000), esto se lo puede observar en la figura 9.

Figura 9

Estructura de red neuronal.



Red neuronal convolucional (CNN)

Es un tipo especial de red neuronal multicapa o arquitectura de aprendizaje profundo que se inspira en el sistema visual de los seres vivos. (Ghosh et al., 2020). Esta red neuronal toma este nombre de la operación matemática lineal entre matrices llamada convolución, CNN tiene varias capas, incluida la capa convolucional, la capa de no linealidad, la capa de agrupación y la capa totalmente conectada (Albawi et al., 2018). La CNN tiene un excelente desempeño en:

- Problemas de aprendizaje automático.
- Aplicaciones que se ocupan de datos de imágenes.
- Visión por computadora.
- Procesamiento de lenguaje natural (NLP).

La tabla 5 detalla las características de los elementos que conforman una CNN y sus detalles.

Tabla 5

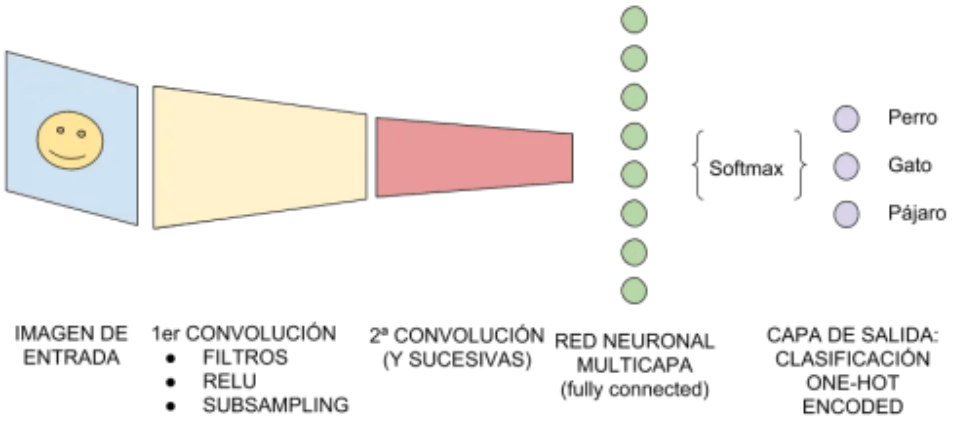
Características de una CNN.

Característica	**Red Neuronal Convolucional**
Datos de entrada	Recibe los pixeles de una imagen. Son tres capas RGB, si la imagen es a color
Capas ocultas	Contienen capas del tipo: Convolucionales, con un tipo de kernel y tamaño de filtros. Capas de Muestro o Subsampling
Capa de salida	Se debe realizar una operación para aplanar la última convolución y agregar una capa Softmax que genere directamente una predicción.
Aprendizaje	Supervisado
Cantidad de capas ocultas	Permiten la detección de características de la imagen mediante una jerarquía, siendo las primeras capas capaces de detectar rasgos como líneas, posteriormente curvas, bordes y progresivamente rasgos más elaborados.
Backpropagation	Empelado para el cálculo de los pesos de los kernels

La figura 10 muestra la arquitectura general de una CNN, para el que se detalló las características fundamentales según Barrios, (2022), detalla el ingreso de la imagen hasta su capa de salida con la predicción.

Figura 10

Arquitectura de una CNN.



Redes neuronales recurrentes (RNN)

Los RNN son redes de neuronas con conexiones de retroalimentación y posiblemente sean biológicamente más plausibles que otros modelos adaptativos. En particular, los RNN pueden usar su estado oculto (memoria) para procesar secuencias de entradas de longitud variable (Lim, 2021).

Transfer Learning

Dentro de ML los datos de entrenamiento y los datos de prueba se toman del mismo dominio, dando como resultado que el espacio de características que tenemos como entrada y las características de distribución de datos son las mismas. Sin embargo, en algunos escenarios del mundo real, esto no se cumple. Existen casos en los que los datos de entrenamiento poseen un alto costo o son difíciles de recopilar o acceder. Por lo tanto, existe la necesidad de crear aprendices de alto rendimiento capacitados con datos más fáciles de obtener de diferentes dominios. Esta metodología se conoce como transferencia de aprendizaje

(Weiss et al., 2016). El aprendizaje por transferencia se utiliza para mejorar a un alumno de un dominio mediante la transferencia de información de un dominio relacionado.

Como estrategia para trabajar con redes neuronales profundas, se emplean redes previamente pre-entrenadas con bases de datos grandes y así poder adaptarlas a un problema específico. Entre ellas existen redes disponibles en Keras que fueron entrenadas con base de datos ImageNet, podemos mencionar:

- Xception
- InceptionV3
- ResNet50
- VGG16
- VGG19
- MobileNet

Según se menciona en Keras, (2022) el flujo de trabajo sugerido para realizar transferencia de aprendizaje consta de cuatro pasos detallados a continuación:

- Tomar capas de un modelo previamente entrenado.
- Congelar las capas para evitar destruir la información que contienen durante futuras rondas de entrenamiento.
- Agregar algunas capas nuevas y entrenables encima de las capas congeladas existentes, estas aprenderán a convertir las características antiguas en predicciones sobre un nuevo conjunto de datos.
- Entrenar las capas nuevas en su conjunto de datos.

ResNet-50

Es una CNN que se entrena con la base de datos ImageNet, la cual tiene más de un millón de imágenes. Esta red se diferencia porque tiene 50 capas de profundidad y puede

clasificar imágenes en 1000 categorías de objetos. El tamaño de entrada de la imagen es de 224 x 224, y como resultado, la red ha aprendido varias representaciones en funciones para una amplia gama de imágenes.

ImageNet

Es una base de datos de imágenes organizada según la jerarquía de WordNet, en la que cada nodo de la jerarquía está representado por cientos y miles de imágenes. El proyecto ha sido fundamental en el avance de la visión por computadora y la investigación de aprendizaje profundo (ImageNet, 2021).

Matriz de confusión

Esta es una herramienta empleada dentro del campo de inteligencia artificial la cual permite evaluar el desempeño de un algoritmo de aprendizaje supervisado. La matriz contiene en sus columnas el número de predicciones de cada clase determinadas por el algoritmo, mientras que en sus filas representa las instancias en la clase real, es decir, pone en relación las predicciones del algoritmo con los resultados correctos que se mostraron o debían haber mostrado.

A continuación se presenta una matriz binaria semejante a la presentada por Santra & Christy, (2012) para ejemplificar los valores que contienen las matrices de confusión.

Actual	**Positivo**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativo	Falsos Positivos (FP)	Verdaderos Negativos (VN)
		Positivo	**Negativo**
		Predicción	

En esta matriz se considera lo siguiente:

- VP es la cantidad de positivos que fueron clasificados correctamente como positivos por el modelo.
- VN es la cantidad de negativos que fueron clasificados correctamente como negativos por el modelo.
- FN es la cantidad de positivos que fueron clasificados incorrectamente como negativos.
- FP es la cantidad de negativos que fueron clasificados incorrectamente como positivos.

Con estos valores se puede determinar parámetros que permiten determinar el desempeño del modelo.

Exactitud (Accuracy)

Determina el porcentaje de los datos clasificados de manera correcta

$$Exactitud = \frac{VP + VN}{Total}$$

Tasa de error (Miss classification Rate)

Determina el porcentaje de los datos clasificados de manera incorrecta

$$Tasa\ de\ error = \frac{FP + FN}{Total}$$

Sensibilidad, exhaustividad (Sensitivity, Recall)

También llamada tasa de verdaderos positivos, determina el porcentaje que se logra clasificar cuando la clase es positiva.

$$Sensibilidad = \frac{VP}{VP + FN}$$

Especificidad (Especifity)

También llamada tasa de verdaderos negativos determina el porcentaje que se logra clasificar cuando la clase es negativa.

$$Especificidad = \frac{VN}{VN + FP}$$

Precisión

Determina el porcentaje clasificado correctamente cuando se predice positivos.

$$Precisión\ = \frac{VP}{VP + FP}$$

Valor de predicción negativo (VPN)

Determina el porcentaje clasificado correctamente cuando se predice negativos.

$$VPN = \frac{VN}{VN + FN}$$

Valor – F (F1-score)

Determina el rendimiento combinado de la precisión y la exhaustividad, mediante la media armónica entre ellas.

$$VF = 2 * \frac{precisión * exhaustividad}{precisión + exhaustividad}$$

Visión por computadora

Es un capo de la inteligencia artificial, el cual permite que las computadoras y los sistemas puedan obtener información relevante de imágenes digitales, videos y otras entradas visuales, en base a los datos obtenidos se pueda hacer recomendaciones basadas en dicha información. Es decir que en el sistema la visión por computadora le permite observar y comprender su entorno. (IBM, 2022)

Servicios Web

Según Subbulakshmi et al., (2020) un servicio en la nube o servicio web, es aquel servicio de software el cual brinda soluciones comerciales consumidas por diferentes solicitantes de servicios y al que se puede acceder mediante un protocolo web estándar. Los servicios manejan el procesamiento en la nube o Cloud Computing, y con ello engloba a todas, las infraestructuras, plataformas, tecnologías o sistemas de software que permiten el acceso de usuarios por medio de internet, entre ellas se mencionan las siguientes:

- Infraestructura como servicio (IaaS): ofrece acceso bajo demanda a recursos de computación, servidores, recursos informáticos, de red y de almacenamiento que pueden ser escalables.
- Plataforma como servicio (PaaS): Incluye infraestructura, pero también incluye middleware, herramientas de desarrollo, servicios de inteligencia empresarial (BI), sistemas de administración de bases de datos, etc.
- Software como servicio (SaaS): entrega aplicaciones de software basadas en la nube, la plataforma en la que se ejecuta y la infraestructura subyacente.
- Función como servicio (FaaS): proporciona a desarrolladores la posibilidad de diseño, ejecución y gestión de paquetes de una pieza lógica o parte de una aplicación, sin tener que preocuparse por el mantenimiento de la infraestructura.

Azure

Es una plataforma informática en la nube de Microsoft, que incluye análisis, redes, cómputo y almacenamiento, mediante estas características los usuarios pueden desarrollar diversas aplicaciones o ejecutar algunas que ya se encuentran en la nube. Esta plataforma incluye Infraestructuras como servicios (Iaas), Software como servicio (SaaS), Plataforma como servicio (PaaS) y serveless que se puede utilizar como un servicio de análisis.

El cual cuenta con ventajas como:

- Es seguro, por lo que permite confiar en la nube.
- Se lo puede integrar sin ningún problema.
- Soporta todos los lenguajes y frameworks.
- Cuenta con actualizaciones innovadoras de Microsoft.

Azure Computer Vision. Es una API de Microsoft, basado en la nube, la cual brinda a los diversos desarrolladores un acceso a algoritmos avanzados para el procesamiento de imágenes y la devolución de información. Cuando se carga una imagen o se especifica la URL de la imagen, los algoritmos con los que cuenta Microsoft Computer Vision pueden analizar el contenido visual de diferentes maneras según las entradas y las elecciones del usuario. (Microsoft, 2022b)

Image Analyze. Es una herramienta la cual puede extraer las características visuales de la imagen. Gracias a esta funcionalidad puede determinar el contexto de la imagen, y después poder categorizarla según las especificaciones del usuario.

Sus principales ventajas de uso son:

- Etiquetar características visuales.
- Detección de objetos.
- Detectar marcas
- Categorización de imágenes.
- Descripción de la imagen.
- Detección de rostros.
- Genera miniaturas
- Detección de colores

Herramientas de software

Django

Es un framework web de Python de alto nivel, el cual fomenta un desarrollo rápido y un diseño limpio y pragmático. Es gratis y de código abierto.(Django, 2022)

Dentro de sus ventajas son:

- Rápido: ayuda a los desarrolladores a llevar sus aplicaciones desde el concepto hasta su finalización lo más pronto posible.
- Seguro: ayuda a los desarrolladores a evitar varios errores de seguridad que son muy comunes.
- Escalable: varios sitios web aprovechan esta capacidad, para poder escalar de forma rápida y flexible.

Firebase

Es una plataforma móvil de desarrollo creada por Google, que tiene como función principal la creación de aplicaciones de una forma rápida y segura. La plataforma se encuentra en la nube y es compatible con iOS, Android y web. Cuenta con diversas funciones para que el desarrollado pueda adaptar esta plataforma a su aplicación, dentro de las características que la destacan son:

- Permite el desarrollo de aplicaciones minimizando el tiempo de optimización y desarrollo.
- Tener un control de rendimiento mediante métricas analíticas.
- Permite gestionar de manera fácil a los usuarios
- Puede adaptarse a diversas aplicaciones

Python

Es uno de los lenguajes de programación más usados hoy en día, debido a que es de alto nivel, orientado a objetos, cuenta con una semántica dinámica por lo que es muy atractivo para el desarrollo rápido de aplicaciones. La sintaxis es fácil de aprender, reduce las líneas de código comparado con otros lenguajes, por lo cual disminuye el costo de mantenimiento del programa. Python fomenta la modularidad del programa y la reutilización del código, eso significa que admite módulos y paquetes de la comunidad. (Python, 2022)

JavaScript

Es un lenguaje de secuencias que le permiten funciones complejas en páginas web, y es una de las tecnologías centrales junto con HTML y CSS, por lo cual es muy usado por los desarrolladores de páginas web.

Bootstrap

Es el framework de HTML, CSS y JavaScript más popular para el desarrollo de un sitio web compatible con diversos dispositivos móviles, es decir ayuda a que los diseños sean responsivos. Una de sus grandes ventajas es que este framework es gratis, fácil y rápido de usar.

Protocolos y técnicas

Web socketcs

Según Melnikov (2011), el protocolo WebSocket permite la comunicación bidireccional entre un cliente que ejecuta un código que no es de confianza en un entorno controlado y un host remoto que ha optado por recibir comunicaciones desde ese código.

Es un protocolo con estado, lo que significa que la conexión entre el cliente y el servidor estará activa hasta que cualquiera de las partes la finalice, una vez finalizada la conexión por alguna de las partes la conexión terminará para ambos, a diferencia de HTTP este comienza desde ws:// o wss://.

WebRTC

Agrega comunicación en tiempo real a diferentes aplicaciones, admite voz, video y datos genéricos que se envíen entre pares, lo que lo hace una herramienta sumamente útil para el desarrollo de soluciones de comunicación de voz y video. Las tecnologías que están detrás de WebRTC se implementan como un estándar abierto y están disponibles como API regulares de JavaScript. El proyecto WebRTC es de código abierto y se encuentra apoyado por grandes empresas como los son: Apple, Google, Microsoft y Mozilla, entre otros.

Ajax

Según Garrett (2005) Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes.

Ajax incorpora:

- Presentación basada en estándares usando XHTML y CSS.
- Visualización dinámica e interacción utilizando el modelo de objeto de documento.
- Intercambio y manipulación de datos mediante XML y XSLT.
- Recuperación de datos asíncrona usando XMLHttpRequest.

Es muy usado por desarrolladores web, debido a que permite mejorar la interacción entre el usuario y la aplicación, evitando recargas constantes en la página, estos intercambios de datos entre el cliente y el servidor se los realiza en segundo plano, todo esto gracias a la creación de la capa intermedia de Ajax entre el usuario y el servidor para mejorar la respuesta de la aplicación.

OpenCv

Es una biblioteca de software de aprendizaje automático y visión artificial de código abierto. Está se creó para poder proporcionar una infraestructura común para las aplicaciones de visión por computadora, consta con más de 2500 algoritmos optimizados. Estos algoritmos

suelen ser usados para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas mediante videos, rastreo de movimiento, producir nube de puntos 3D, etc.

Gracias a todas estas características se desarrollan diversos trabajos como monitoreo de maquinaria, detección de intrusos en videos de vigilancias, asistencia en robots, detección de accidentes y más. Cuenta con interfaces C++, Python, Java y M de Matlab y es compatible con Windows, Linux, Android y Mac OS. El uso de OpenCV se inclina principalmente hacia aplicaciones de visión en tiempo real.(OpenCV, 2022)

Tensorflow

Es una plataforma de código abierto integral para el aprendizaje automático. Tiene un ecosistema completo y flexible de herramientas, bibliotecas y recursos de la comunidad que permite a los investigadores impulsar lo último en aprendizaje automático y a los desarrolladores crear e implementar fácilmente aplicaciones basadas en aprendizaje automático.

Esta plataforma da grandes ventajas como:

- Fácil construcción de modelos.
- Producción robusta de ML en cualquier lugar.
- Potente experimentación para la investigación.

Heroku

Es una plataforma en la nube, la cual soporta varios lenguajes de programación como Go, PHP, Java, Python, entre otros. Heroku permite monitorear y escalar aplicaciones, es una forma rápida de poder subir una aplicación a un servidor y de esta manera poder acceder a la página desde cualquier dispositivo con conexión a Internet.

Hardware

Para el desarrollo del proyecto hay que tener en claro los conceptos de los elementos que se describen en la tabla 6.

Tabla 6

Elementos de Hardware.

Elemento	Descripción
Raspberry Pi	Es una computadora de bajo costo, que se conecta a un monitor o TV. Este dispositivo permite programar en lenguajes de programación como Scratch y Python
Servomotor	Es un actuador rotativo que sirve como un dispositivo de accionamiento para un control de posición, precisión de velocidad y aceleración
Sensor infrarrojo	Es un dispositivo optoelectrónico capaz de medir la radiación electromagnética infrarroja de los cuerpos en su campo de visión, sin importar la cantidad de luz que exista o las condiciones ambientales en las que se encuentre.
Cámara	Es un dispositivo óptico utilizado para capturar imágenes o fotografías
Motor DC	Permite transformar la energía eléctrica en energía mecánica, el cual provoca un movimiento rotatorio gracias a los campos magnéticos que se crean.

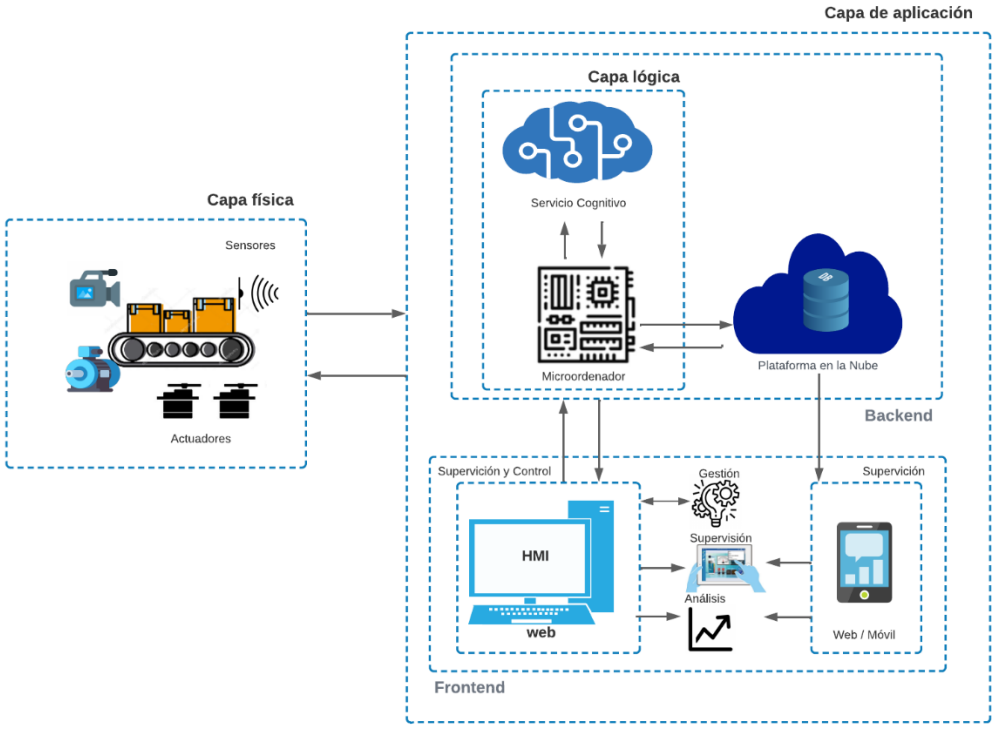
Capítulo 3: Diseño

Diseño de arquitectura

Para el diseño de la arquitectura del sistema se procuró seguir el modelo mencionado en el capítulo uno y ampliada en la figura 11, es decir contendrá una capa a nivel físico, una capa a nivel lógico y una capa de aplicación. El diseño a nivel físico contiene la selección del tipo de estructura mecánica y sus limitaciones en dimensiones para el ingreso de objetos al proceso de clasificación, además de la selección del tipo de sensores y actuadores para el acople en la estructura seleccionada. Ya a nivel lógico se detalla la metodología empleada y el diseño del algoritmo principal del sistema clasificador, mismo que comandará cada elemento del sistema y permitirá su conexión hacia la nube para el consumo del servicio web cognitivo elegido. Finalmente en el nivel de aplicación se aborda el diseño de la interfaz de control y de supervisión, siendo necesario dividir este en dos capas o dos niveles, el frontend y el backend, el primero contendrá las metodologías y técnicas de diseño que permitan al usuario de la aplicación una interacción sencilla con el sistema clasificador, mientras que la segunda implementará y manejará la lógica de la aplicación, esta se basará en el algoritmo central desarrollado en el nivel lógico de la arquitectura, en otras palabras, permitirá que el sistema funcione de manera correcta mediante los datos ingresados por el usuario en la interfaz.

Figura 11

Arquitectura del sistema.



Requisitos de la Arquitectura

Se establece la tabla 7 como guía de diseño, en la que se plantea los requisitos que deberán cumplir los elementos dentro de cada nivel de la arquitectura del sistema clasificador.

Tabla 7

Requisitos de Arquitectura.

Requisito	Descripción
Capa física	
Estructura mecánica	• Permitirá al usuario el ingreso de objetos de manera serial, uno detrás de otro.

Requisito	Descripción
	• Será factible el acoplamiento de sensores y actuadores electrónicos. • Simula un proceso en una industria, por lo que, se deberá asemejar a dichas estructuras. • Sus sensores y actuadores deben poder ser comandados por un microordenador.
Capa Lógica	
Algoritmo principal	• Permite la identificación de objetos y clasificación en dos grupos determinados por el usuario. • Basa su funcionamiento en el consumo de un servicio web cognitivo. • Permite el aprendizaje de objetos no identificados por el servicio cognitivo. • Evita el consumo excesivo de recursos en el microordenador aprovechando las funcionalidades del procesamiento en la nube.
Capa de aplicación	
Frontend	• Se generan dos interfaces, una interfaz web que permite el control y supervisión del proceso en tiempo real, y una interfaz web/móvil para supervisión del estado del proceso en tiempo real.

Requisito	Descripción
Backend	• El diseño de cada interfaz divide de manera clara cada función que posea. • Permite la implementación del algoritmo principal, así como la conexión a una base de datos en la nube. • Maneja los lenguajes de programación necesarios para poder identificar y clasificar objetos, así como comandar sensores y actuadores de la estructura, además de enviar datos a la base de datos en la nube y comunicarse con la interfaz de usuario.

Para dar inicio al diseño se empieza con el nivel lógico pues contiene la esencia del sistema, en él se decide los parámetros de diseño del algoritmo principal, posteriormente se detalla el nivel de aplicación pues en él explicaremos como implementar el algoritmo antes diseñado y unirlo a un interfaz de usuario, por último, se diseña el nivel físico en el que se plasma de manera real un proceso de clasificación.

Capa Lógica

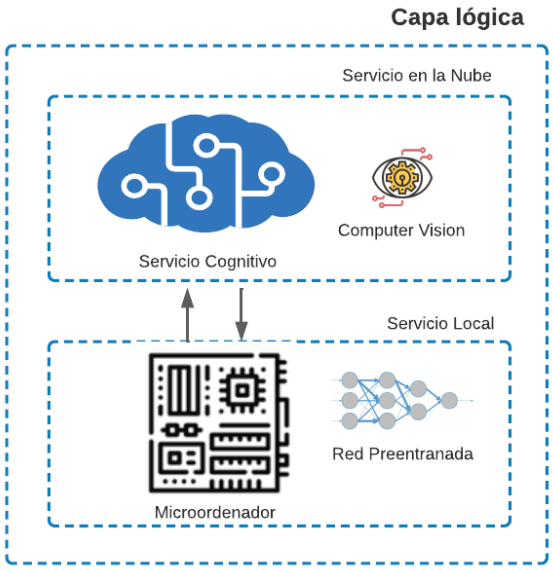
El sistema posee dos características bien definidas, la primera es la identificación de objetos mediante el consumo de un servicio web cognitivo, y la segunda es la de aprendizaje de nuevos elementos que no sean posible identificarlos mediante el servicio web, para posteriormente clasificarlos en dos clases determinadas por el usuario del sistema. Para cumplir con el primer requerimiento será necesario la comparación de los diferentes proveedores de servicios existentes en la web, entre ellos Microsoft, Amazon, Google, e

identificar sus herramientas de visión, para el caso de Microsoft se analiza Azure Cognitive Services, para Amazon será AWS Rekognition y para Google Vision AI, cada uno de ellos ofrece sus servicios en forma de API REST, que facilita el uso de inteligencia artificial dentro del desarrollo de aplicaciones, la llamada de dichas APIs permiten la identificación y etiquetado de objetos dentro de imágenes.

Para la segunda característica de aprendizaje y generación de un modelo que permita el ingreso de nuevos objetos al sistema, existen ciertas limitaciones, pues el consumo de servicios web permiten el uso de Softwares como servicio, y el acceso al código se limita, y con ello la funcionalidad que para este proyecto se desea dar. Por ello se decide la creación de un modelo de aprendizaje mediante una red preentrenada que correrá dentro del microordenador, aprovechando la característica de transferencia de aprendizaje de la red, evitando elevar los requerimientos de hardware de la tarjeta.

Figura 12

Arquitectura de capa lógica.



Así, la figura 12 muestra que el algoritmo del sistema basará su funcionamiento en dos servicios, un servicio en la nube que permita la identificación de objetos con las características, ventajas y desventajas que el proveedor del servicio web cognitivo brinde, y un servicio local que facilite el aprendizaje e identificación de nuevos objetos mediante un modelo preentrenado.

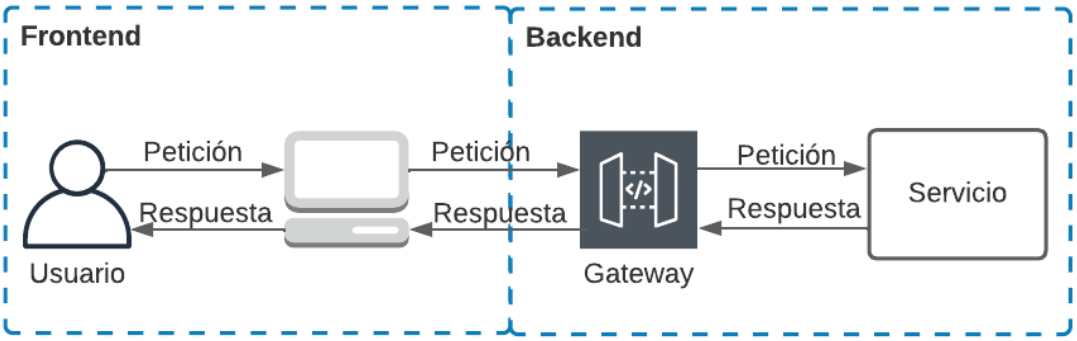
Capa de Aplicación

En esta capa se usan las características descritas a nivel lógico, dentro de la capa de aplicación se desarrollan las interfaces las cuales se dividen en dos partes principales, la interfaz local y la interfaz remota, esta división permite controlar el ingreso de usuarios a la interfaz local, evitando múltiples solicitudes de comandos de control al sistema, y provocar daños irreparables en los equipos, también ayuda a mejorar la seguridad del sistema en donde solo personal autorizado pueda acceder a los controles de este.

El desarrollo de esta capa es fundamental, debido a que se conecta con la capa física para la comunicación de los datos existentes en el sistema, y de esta manera mostrar la información al usuario, por lo cual, tanto para la interfaz local como para la remota, es necesario la división del software en dos etapas, la primera es el Backend y la segunda es el Frontend, esta división permite comprender de mejor manera el desarrollo del software mediante una correcta estructura, la figura 13 ilustra el diagrama de conexión entre el Frontend y Backend, estas dos capas se deben conectar entre sí para poder comunicar y procesar los datos requeridos por el usuario.

Figura 13

Diagrama de conexión Frontend con Backend.



Interfaz Local

Para el acceso a esta interfaz se lo realiza a través del ingreso directo al microordenador, de esta manera se restringe el ingreso al control del sistema, la interfaz muestra al usuario datos en tiempo real y obtiene el manejo del sistema, en el cual el operador puede seleccionar los objetos, activar, detener, pausar y más opciones, es decir la funcionalidad de esta interfaz es de control y supervisión.

El acceso a un sistema de control debe ser limitado, un acceso público puede tender a generar fallas en los sistemas al recibir varias peticiones de manejo a la vez, dando como resultado averías en los elementos conectados.

Interfaz Remota

El acceso a esta interfaz es público y se lo puede hacer desde cualquier dispositivo con conexión a Internet al ingresar a una URL específica, esta interfaz a diferencia de la local no se puede acceder a los comandos de funcionalidad, únicamente se logra visualizar los datos obtenidos del sistema en tiempo real.

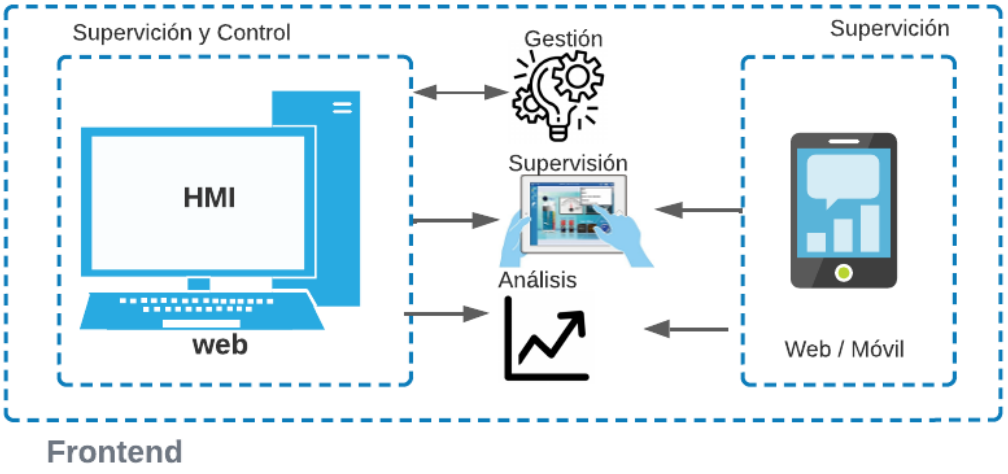
Frontend

Es una parte principal en el desarrollo del proyecto, esta capa permite las peticiones por parte del usuario, y en base a dichas solicitudes se obtendrá una respuesta por parte del Backend, por lo que, la distribución de los elementos en la interfaz debe ser adecuada, para obtener un manejo sencillo, cómodo y fluido.

En la figura 14 se puede observar los permisos que brinda esta capa a los usuarios, donde por medio de una interfaz local se tiene acceso a la gestión, supervisión y análisis de los datos del sistema, y la interfaz remota cuenta únicamente con acceso a supervisión y análisis, todos estos comandos son ingresados al Backend, en donde se procesan y se ejecutan diversas funciones para el cumplimiento de los requerimientos del usuario.

Figura 14

Estructura del Frontend.

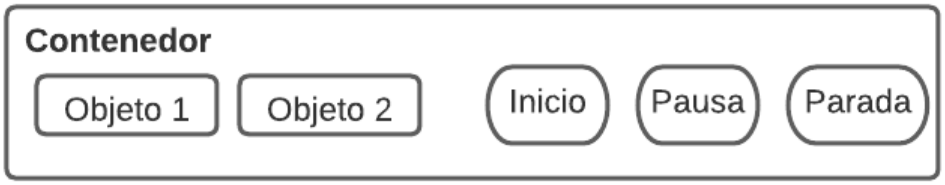


Funcionamiento de los componentes de control. Cada componente cuenta con una funcionalidad distinta, y su propósito es realizar una función en específico que solicite el cliente, estos componentes se comunican con el Backend en donde se realiza las consultas

respectivas para la obtención de datos necesarios y poder realizar la función especificada, en la figura 15 se muestra un ejemplo de la distribución de los botones de control.

Figura 15

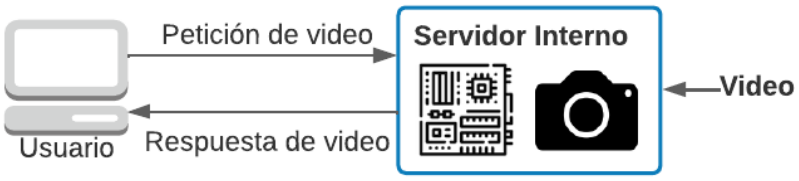
Diseño de contenedor.



Componente de video de la aplicación. Este componente básicamente ofrece la transmisión de video en tiempo real de la cámara conectada al microordenador, por lo que es necesario el uso de una tecnología que permita la comunicación al instante, la figura 16 muestra el diagrama de cómo se transmite el video a un dispositivo.

Figura 16

Transmisión de datos.



Backend

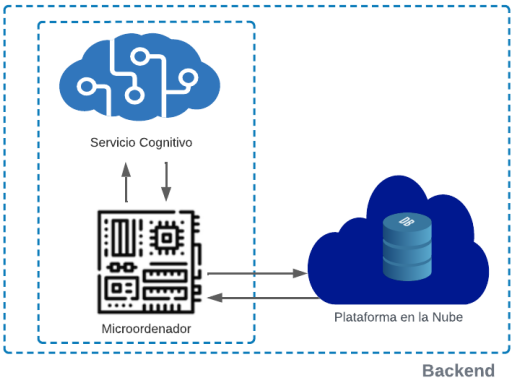
A diferencia del Frontend esta capa se vuelve el núcleo de la arquitectura, convirtiéndose en una parte fundamental en el desarrollo del proyecto, las funciones que desempeña esta capa son importantes debido a que se encarga de la gestión de los datos pedidos por el usuario y los envía al servidor, almacena la información en una base de datos permitiendo que estén seguros, y también incorpora librerías y configuraciones necesarias para

el correcto funcionamiento del sistema. Esta capa contiene el servicio local y el servicio en la nube que se ofrece al usuario, por ello, es importante al momento de enlazarse con el Frontend observar las funcionalidades que pueden ser ejecutadas desde el Backend.

Estructura general del Backend. En esta estructura se encuentran elementos importantes para el correcto funcionamiento del proyecto, la figura 17 muestra las características que tiene el microordenador con sus conexiones a diferentes servicios en la nube, en esta capa el usuario no tiene acceso a realizar algún cambio tanto en el código como los enlaces a la nube, esta comienza a trabajar una vez que el usuario solicite algún requerimiento desde la capa de Frontend, por lo que es importante tener un correcto enlace entre la capa Backend y Frontend.

Figura 17

Estructura del Backend.



Capa Física

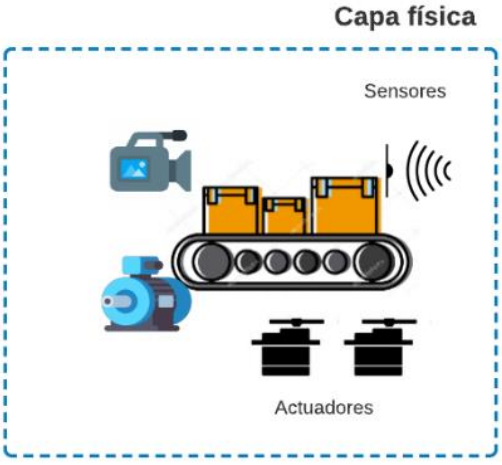
Para demostrar el funcionamiento de la arquitectura implementada, es necesario el desarrollo de un medio físico, el cual incorpore todo lo planteado anteriormente, este debe contar con los sensores adecuados para la toma de datos, y con los actuadores que cumplan con tareas específicas, es importante conocer que el accionamiento de cada actuador en el proceso será especificado desde la capa Frontend por el usuario, posteriormente esa

información es captada en el Backend para realizar las respectivas activaciones y configuraciones y de esta forma poder manipular el componente.

Uno de los sistemas de transporte de materiales dentro de la industria que resulta eficiente y ha abarcado en gran medida el sector, son las bandas transportadoras (Basurto, 2013), por lo que, se opta por la simulación a escala de esta estructura, esta permite el transporte en serie de los objetos, y será dimensionada para elementos de tamaño reducido, como frutas, ciertos vegetales, y objetos de peso y tamaño semejantes a una manzana como referencia. Para generar el movimiento de la banda se acoplará un motor eléctrico al mecanismo de giro, como actuadores que permitan al sistema la clasificación en dos grupos, se eligen servomotores con palancas adaptadas como compuertas que permitirán el paso o no de los objetos. Finalmente, para la captación de datos, será necesario el uso de sensores de presencia para la detección del paso de los objetos, y por otro lado dos cámaras, una para capturar la imagen del objeto cuando el sensor de presencia lo detecte y otra que permita realizar un stream de todo el proceso y mostrarlo en la interfaz de control, la figura 18 ilustra de manera simple los elementos principales de la arquitectura en la capa a nivel físico.

Figura 18

Estructura de capa física.



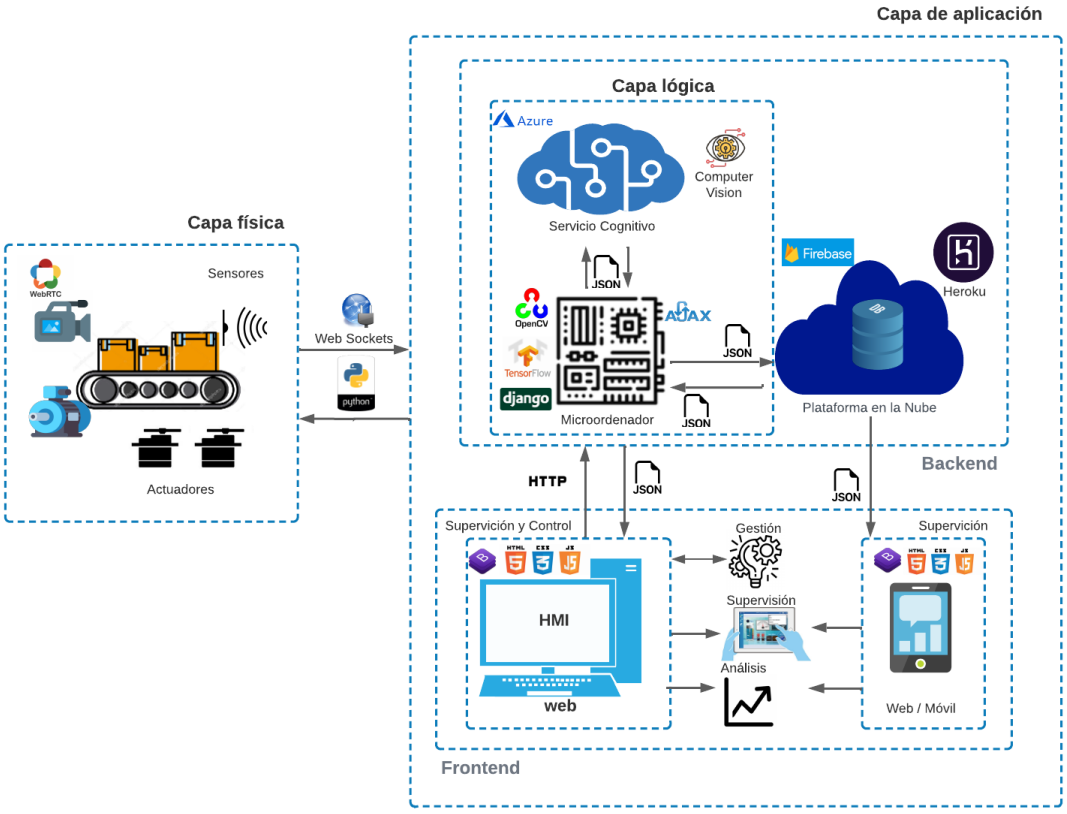
Capítulo 4: Implementación

Estructura general de la arquitectura

Una vez planteado los requerimientos del diseño en cada una de las capas de la arquitectura, se procede a definir las herramientas y tecnologías necesarias para la implementación del sistema de clasificación. La figura 19, ilustra de manera gráfica y simplificada las herramientas empleadas en cada capa y las tecnologías o protocolos para la comunicación entre ellas.

Figura 19

Arquitectura detallada.



Capa Lógica

Servicio en la Nube

Para dar inicio al proceso de implementación a nivel de la capa lógica, es necesario determinar el servicio cognitivo y su herramienta de visión por computadora que se usará, así como el proveedor de dicho servicio. Se opta por el uso de Microsoft como proveedor y Azure como servidor de la API de Computer Vision.

Computer Vision. Se emplea la API basada en la nube, esta brinda el acceso a algoritmos de extracción de información de imágenes, videos y poder categorizarlos, mediante análisis de imágenes, etiquetado, reconocimiento de rostros, extracción de texto y análisis espacial, dando información sobre sus características visuales. El servicio puede brindar análisis de varias formas según el tipo de entrada y la elección del desarrollador. La API de Computer Vision basa su funcionamiento en el uso de redes neuronales profundas y más específicamente en una Red Neuronal Convolucional, este permite al sistema de inteligencia artificial convertir las imágenes de entrada en matrices digitales, y se emplean para la clasificación de imágenes, detección y reconocimientos de objetos, rostros y otras aplicaciones más. Para iniciar su uso se consideran las siguientes características del servicio.

Creación de un Recurso de Cognitve Services. Para la creación de un recurso Microsoft permite el uso de los siguientes métodos:

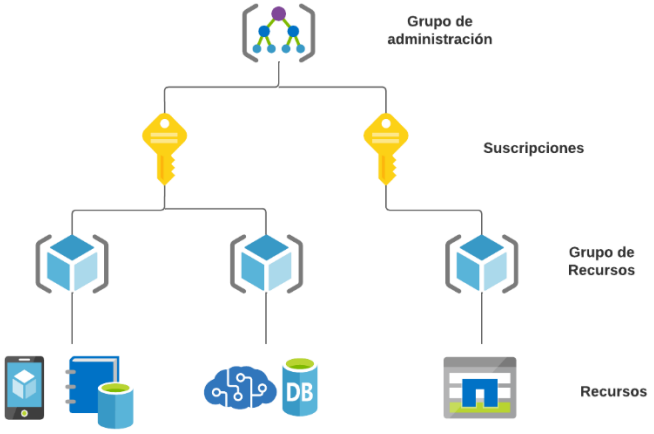
- Portal azul
- CLI de Azure
- Bibliotecas de cliente de Azure SDK
- Azure Resource Manager (plantilla ARM)

Seleccionamos el primer método, para el cual será necesario la creación de una cuenta en el Portal web de Azure. Dentro de este, el grupo de recursos será el contenedor que almacena los recursos relacionados con una solución de Azure, este grupo contiene y

almacena metadatos acerca de cada recurso, siendo importante asegurarnos que los datos se almacenen en una región concreta. Previo al uso de la plataforma, es necesario tener claro la estructura manejada por Azure e ilustrada en la figura 20, en la que es necesario la creación de una suscripción para el acceso a un grupo de recursos y en este administrar cada recurso de manera individual, para este proyecto, será el servicio cognitivo de Computer Vision.

Figura 20

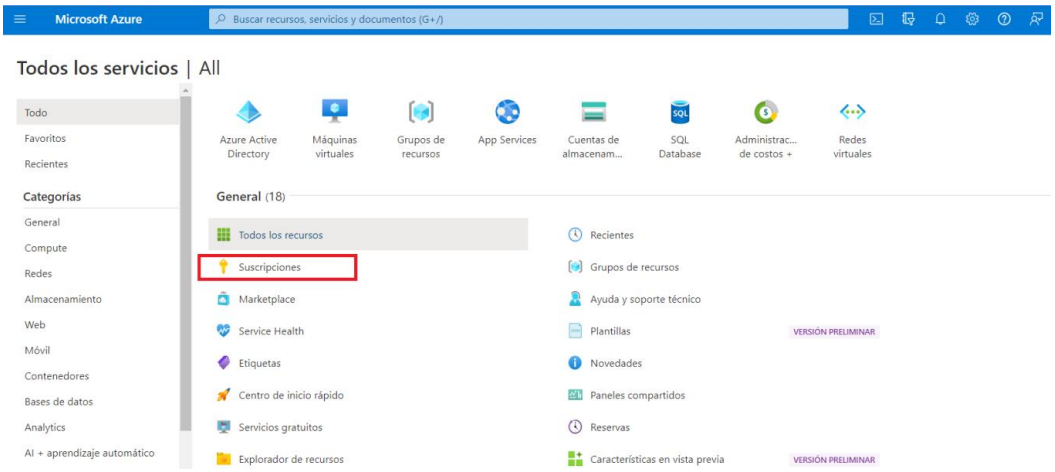
Estructura de Azure.



Previo a la creación del grupo de recursos, y una vez dentro del Portal de Azure, se genera una suscripción, la figura 21 ilustra los recursos existentes en el portal y la selección de las suscripciones.

Figura 21

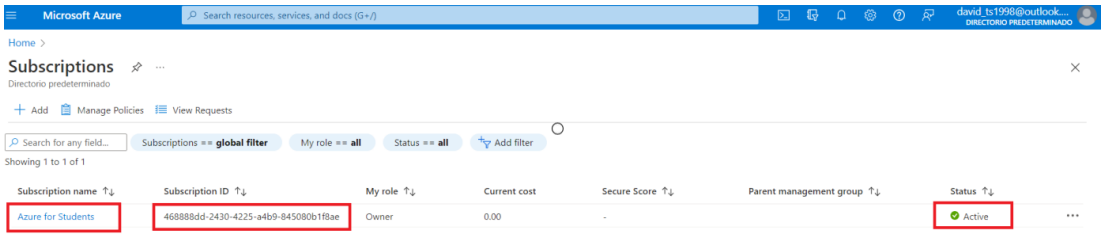
Pantalla de recursos de Azure.



Para el proyecto se empela una cuenta con suscripción de estudiante F0, obteniendo los beneficios y limites gratuitos que brinda la plataforma. La figura 22 muestra el tipo de suscripción, ID, costo y el estado de dicha suscripción la cual habilitará el acceso al uso de recursos.

Figura 22

Pantalla de suscripciones de Azure.



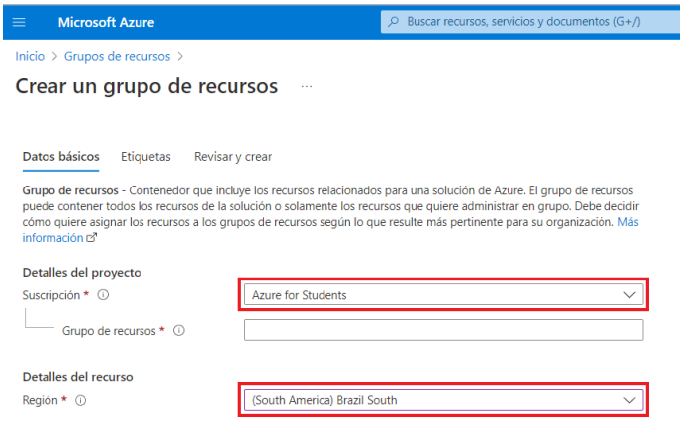
Una vez generado la suscripción se deberá crear un grupo de recursos el cual contendrá y administrará cada uno de los recursos individuales que Azure dispone y que el tipo de suscripción creada lo permita y le de acceso. Los parámetros importantes para la creación

del grupo serán el tipo de suscripción, el nombre del recurso y la región, considerando que la API que se usará está disponible en varios sectores del mundo, y para el proyecto se seleccionó el servidor más cercano mostrado en la figura 23, para el caso:

- Sur de Brasil: brazilsouth.api.cognitive.microsoft.com

Figura 23

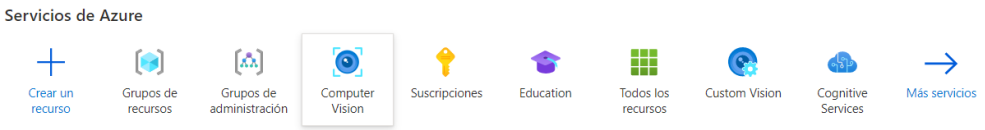
Pantalla de recursos.



Ya creado la suscripción y un grupo de recursos se selecciona el servicio o recursos que se consumirá. La figura 23, muestra la selección del servicio de computer vision.

Figura 24

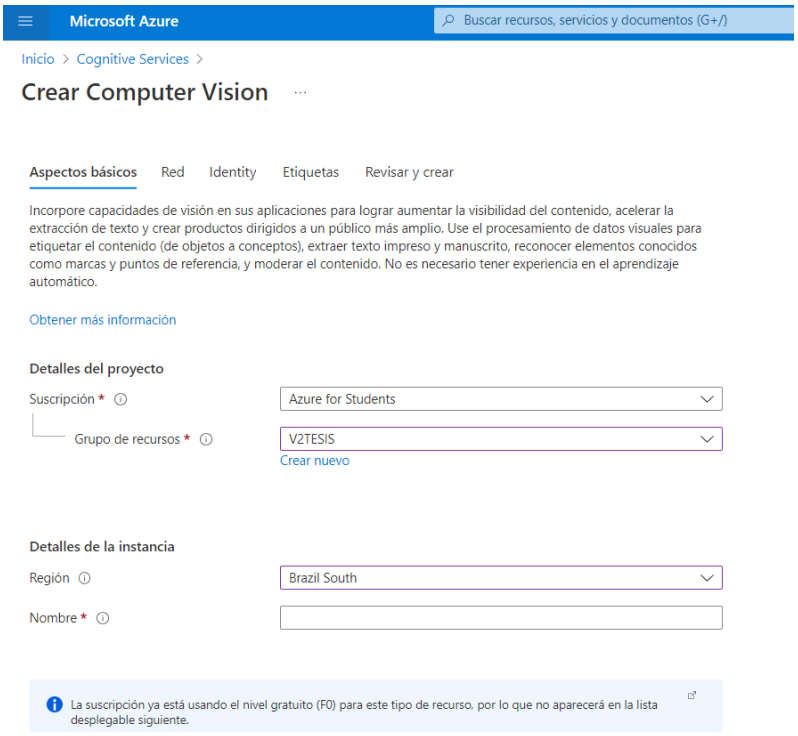
Barra de servicios de Azure.



Para la creación del recurso de Computer Vision se emplean las suscripción y grupo de recursos previamente creados. La figura 25 muestra los parámetros empleados para el uso del servicio.

Figura 25

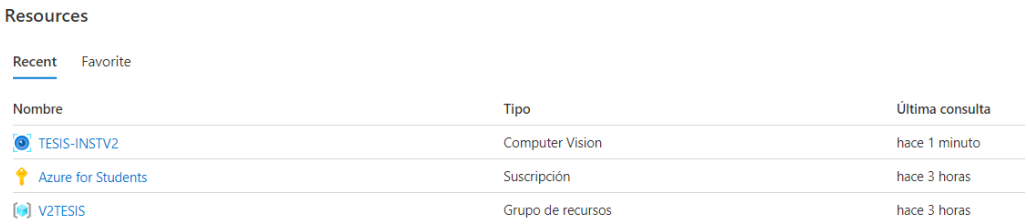
Pantalla de creación de proyecto en Computer Vision.



Se muestra en la figura 26 las herramientas necesarias para el manejo del servicio de Computer Vision, la suscripción el grupo y el servicio o recurso.

Figura 26

Herramientas de Computer Vision.

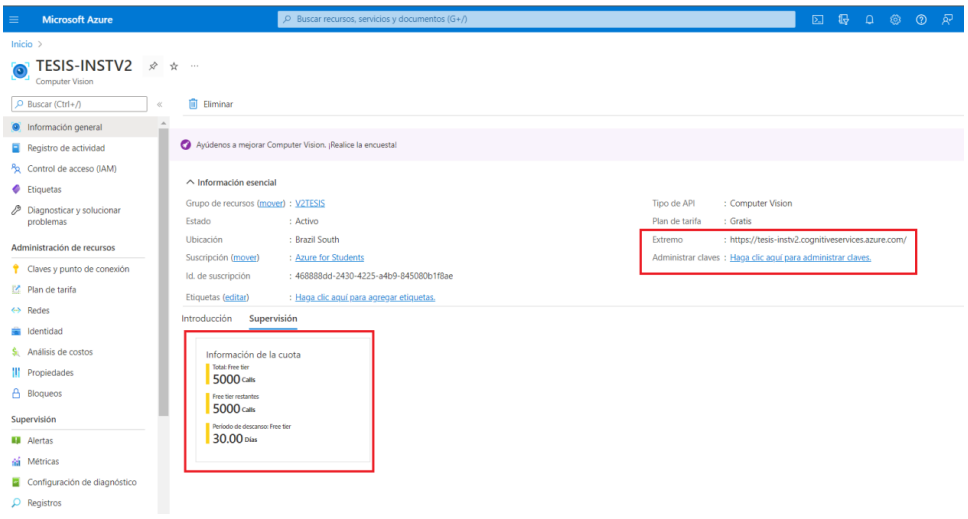


Ya creado el recurso de Computer Vision, en la figura 27 es de importancia identificar el extremo o ENDPOINT y las claves o KEYS que se genera en el administrador de claves, pues estas serán requeridas para el llamado a la API, por otro lado, también podremos supervisar el

consumo del servicio en el Portal de Azure se muestra la información de la cuenta que para el tipo de suscripción se da 5000 transacciones por un periodo de 30 días del servicio web.

Figura 27

Pantalla de información del servicio.



Operación Tag_Image. Emplearemos la operación tag_image como FaaS o función como servicio para acceder a la API de Computer Vision, la operación genera una lista de palabras o etiquetas relevantes según el contenido de la imagen que sea ingresada. Se admiten dos métodos de entrada: el primero mediante la carga de una imagen o el segundo al especificar una URL de imagen. Se devolverá una respuesta exitosa en JSON. Si la solicitud falló, la respuesta contendrá un código de error y un mensaje.

Método HTTP. Se envía por medio del método POST una imagen con los siguientes requisitos de entrada:

- Formatos de imagen: JPEG, PNG, GIF, BMP.
- El tamaño del archivo de imagen debe ser inferior a 4 MB.
- Las dimensiones de la imagen deben ser superiores a 50 x 50.

Para el lenguaje de programación Python la figura 28 muestra el llamado a la operación de tag_image con los parámetros requeridos para su funcionamiento.

Figura 28

Función tag_image.

```
tag_image_in_stream(image, language='en',
model_version='latest', custom_headers=None, raw=False,
callback=None, **operation_config)
```

Nota: Tomado de Microsoft, https://docs.microsoft.com/en-us/python/api/azure-cognitiveservices-vision-computervision/azure.cognitiveservices.vision.computervision.operations.computervisionclientoperationsmixin?view=azure-python#azure-cognitiveservices-vision-computervision-operations-computervisionclientoperationsmixin-tag-image-in-stream

La tabla 8 muestra los parámetros de entrada requeridos para la operación, el tipo de valor del parámetro y sus definiciones o funcionalidades.

Tabla 8

Parámetros de la función.

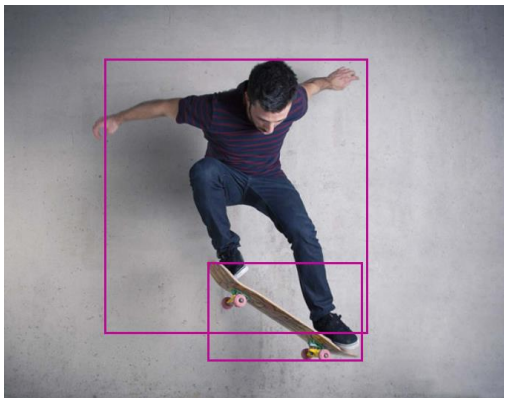
Parámetro	Característica	Tipo	Definición
url	Requerido	str	URL pública de una imagen
lenguaje	Valor por defecto: en	str	Idioma que obtendrá la salida, admite en - inglés, predeterminado. es - español, ja - japonés, pt - portugués, zh - chino simplificado. Los valores posibles incluyen: 'en', 'es', 'ja', 'pt', 'zh'

Parámetro	Característica	Tipo	Definición
model_version	Valor por defecto: último	str	Parámetro opcional para especificar la versión del modelo de IA. Los valores aceptados son: "más reciente", "2021-04-01". El valor predeterminado es "más reciente".
custom_headers	valor por defecto: Ninguno	dic	encabezados que se agregarán a la solicitud
raw	valor predeterminado: falso	Bool	devuelve la respuesta directa junto con la respuesta de serializada
operation_config	Requerido		Anulaciones de configuración de operación.

Los parámetros mencionados y su uso dependerán del desarrollador del proyecto, a excepción de los que son de características requeridas, para el presente proyecto se considera el uso de la operación en lenguaje español, y el primer método de ingreso del parámetro de entrada, es decir con la carga de una imagen y no la de una url pública. Se muestra el ejemplo ilustrado por Microsoft, (2022) en el que se tiene como imagen de entrada la figura 29 y se obtiene un resultado en formato JSON como el mostrado a continuación en la tabla 9:

Figura 29

Resultado de análisis de red.



Nota: Tomado de Computer Vision, por Azure, https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/#overview

Tabla 9

Resultados de la predicción.

Función	Valor
Objetos	[{"rectangle": { "x": 238, "y": 299, "w": 177, "h": 117}, "object": "Skateboard", "confidence": 0.903}, {"rectangle": {"x": 118, "y": 63, "w": 305, "h": 321}, "object": "person", "confidence": 0.955}]
Etiquetas	[{ "name": "skating", "confidence": 0.999951363 }, { "name": "snowboarding", "confidence": 0.9893889 }, { "name": "sports equipment", "confidence": 0.9722208 }, { "name": "person", "confidence": 0.959769964 }, { "name": "roller skating", "confidence": 0.946092963 }, { "name": "skiing", "confidence": 0.92313683 }, { "name": "man", "confidence": 0.9193816 }, { "name": "outdoor", "confidence": 0.9109124 }, { "name": "boardsport", "confidence": 0.907244742 }, { "name": "riding", "confidence": 0.8984571 }, { "name": "sport", "confidence": 0.871290743 }, { "name": "footwear", "confidence": 0.862546742 }, { "name": "snowboard", "confidence": 0.8349905 }, { "name": "skate", "confidence": 0.801233232 }, { "name": "skateboarder", "confidence": 0.792592764 }, { "name": "individual sports", "confidence": 0.779822469 }, { "name": "skateboarding equipment", "confidence": 0.777853966 }, { "name": "skateboard", "confidence": 0.746669054 }, { "name": "skateboarding", "confidence": 0.7466688 }, { "name":

Función	Valor
	"ski", "confidence": 0.6588002 }, { "name": "jumping", "confidence": 0.645534158 }, { "name": "extreme sport", "confidence": 0.5737016 }, { "name": "kickflip", "confidence": 0.501751363 }, { "name": "male", "confidence": 0.15158996 }]
Formato	“Jpeg”

Se presentaron las funciones de relevancia como son, los objetos, etiquetas y formato detectado por el servicio, y dentro de las etiquetas su precisión de detección y predicción, sin embargo, es posible extraer datos como, las dimensiones de la imagen, detección de rostros, el color de fondo dominante y más, según la FaaS que se emplee.

Con cada una de las consideraciones mencionadas en el apartado del servidor en la nube, el servicio está listo para su uso en la capa de aplicación en la sección del Backend.

Servicio Local

La capacidad de aprendizaje de objetos nuevos, no la podrá brindar el servicio de Computer Vision de Azure o como se le llama en el presente proyecto el Servicio en la Nube, por lo que, para conseguir dar esta funcionalidad al sistema clasificador se opta por el uso de ResNet50 como red preentrenada que permitirá emplear la técnica de transferencia de aprendizaje. El aprendizaje por transferencia generalmente se lo emplea para tareas en las que su conjunto de datos es pequeño para entrenar un modelo a gran escala desde cero, para el caso del sistema de clasificación, esta característica será de gran ayuda pues el usuario deberá ingresar o capturar fotos de los objetos nuevos que desea que el sistema clasifique, con una cantidad de 20 imágenes por objeto.

Previo a la implementación de la red y su entrenamiento, es necesario tener claro que las librerías mostradas por la figura 30 son necesarias para conseguirlo, tensorflow para el llamado de keras, Dense, Sequential, Flaten como capas que serán añadidas.

Figura 30

Librerías para Resnet-50.

```
import tensorflow as tf
from tensorflow.python.keras.layers import Dense, Flatten
from tensorflow.python.keras.models import Sequential
import pathlib
```

Por otro lado, también será necesario la generación de data sets con los nuevos datos o imágenes para la nueva red, la figura 31 muestra la generación de dicho set de datos, en el que se especifica los subconjuntos de los datos a devolver de "entrenamiento" y en la figura 32 de "validación", en él se separa el 20% del total de imágenes para validación

Figura 31

Datos de entrenamiento.

```
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
        data_dir,
        validation_split=0.2,
        subset="training",
        seed=123,
        label_mode='categorical',
        image_size=(img_height, img_width),
        batch_size=batch_size)
```

Figura 32

Datos de validación.

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
        data_dir,
        validation_split=0.2,
        subset="validation",
        seed=123,
        label_mode='categorical',
        image_size=(img_height, img_width),
        batch_size=batch_size)
```

Empleamos la clase Sequential para generar la estructura de datos principal en Keras, con ella creamos una red neuronal básica con un modelo secuencial. Obtenemos una secuencia de capas y cada una de ellas irá filtrando gradualmente los datos de entrada para obtener a la salida el valor deseado. Una vez listo la configuración de librerías y del conjunto de datos a emplear, se da inicio al flujo de trabajo recomendado por Keras para tareas que empleen transferencia de aprendizaje:

1. Crear una instancia de un modelo base, y cargar los pesos previamente entrenados.

Como se especificó anteriormente, debido a su gran desempeño en labores de visión por computadora se llama a la red neuronal ResNet50, por medio de la librería keras de tensorflow, como lo ilustra la figura 33, además se detallan en la tabla 10, los parámetros necesarios para su configuración y uso en esta aplicación.

Figura 33

Diseño de red neuronal.

```
print("---------Entrenamiento-----------")
    resnet_model = Sequential()
    pretrained_model =
tf.keras.applications.ResNet50(include_top=False, input_shape=(
        img_height, img_width, 3), pooling='avg', weights='imagenet')
```

Tabla 10

Parámetros de configuración de red Resnet-50.

Parámetro	Descripción
Include_top	Incluye la capa completamente conectada en la parte superior de la red, para el caso actual será False

Parámetro	Descripción
input_shape	Se especifica solo si include_top es falso, contine una tupla con el formato de datos, debe tener 3 canales de entrada, el ancho y altura de la imagen no deben ser menores que 32.
pooling	El modo de agrupación se incluye si include_top es falso y pueden ser: • None: significa que la salida del modelo será la salida del tensor 4D del último bloque convolucional. • Avg: significa que la agrupación promedio global se aplicará a la salida del último bloque convolucional y, por lo tanto, la salida del modelo será un tensor 2D. • Max: significa que se aplicará la agrupación máxima global.
weights	Al ser None se da una inicialización aleatoria, al ser imagenet, carga los pesos del entrenamiento previo en ImageNet

2. Congelar todas las capas en el modelo base configurado en el paso anterior.

Mediante el comando trainable=false el modelo congelará el valor de las capas durante el entrenamiento provocando que sus pesos no se modifiquen cuando se entrena con fit() o cuando se entrena con cualquier bucle personalizado que dependa de trainable_weights, para ello hacemos un barrido por cada capa con un ciclo for y congelamos dichas capas en su último valor. La figura 34 muestra el código para este proceso.

Figura 34

Congelación de capas.

```
for layer in pretrained_model.layers:
        layer.trainable = False
```

3. Crear un nuevo modelo de una o varias capas sobre el modelo base.

Una vez creado el modelo sequential, se pude definir capas empelando el método add(), por lo que, se inicia agregando el modelo preentrenado de ResNet50, posteriormente se añade una capa de aplanamiento Flatten la cual se emplea para convertir los datos multidimensionales en un vector de características 1D para ser utilizado por la siguiente capa que es la capa densa, la capa oculta tiene como función de activación a ReLu, y un número de 512 nodos, para generar una precisión elevada, aunque sacrificando el tiempo de entrenamiento pues este será mayor.

Finalmente, como función de activación de la capa de salida empelamos softmax la cual permite genera un vector de valores que suman 1 y pueden ser interpretadas como la probabilidad de pertenencia a una determinada clase, para el sistema clasificador serán necesarias dos clases como salidas. La figura 35 muestra el código empleado para añadir cada una de las capas antes mencionadas, y dar así con nuestra red neuronal final.

Figura 35

Capas de la red.

```
resnet_model.add(pretrained_model)
    resnet_model.add(Flatten())
    resnet_model.add(Dense(512, activation='relu'))
    resnet_model.add(Dense(2, activation='softmax'))
```

4. Entrenar el modelo creado con el conjunto de datos nuevo.

Partiendo de nuestro modelo nuevo ya definido, se debe configurar su proceso de aprendizaje y cómo se lo realizará, para ello se toma como herramienta el método compile( ), con el que se especifica algunas propiedades a través de los argumentos del método:

- Optimizador: se especifica un optimizador que es la manera de elegir el algoritmo de optimización que permite a la red neuronal calcular los pesos de los

parámetros a partir de los datos de entrada y de la función de loss definida, para el caso será adm.

- Función de perdida: la función de perdida loss permite evaluar el grado de error entre salidas calculadas y las salidas deseadas de los datos de entrenamiento, en el proyecto se emplea categorical_crossentropy como función de loss, provocando que la salida sea de orden categórico, en otras palabras, la variable de salida debe tomar un valor entre las 2 posibles clases definidas.
- Métricas: la métrica empleada para monitorizar el proceso de aprendizaje y prueba de la red neuronal es accuracy la cual usa la fracción de imágenes que son correctamente clasificadas. Keras coge muchos valores por defecto, si no se especifica el argumento validation_data, Keras usa el argumento validation_split, que es un entero entre 0 y 1 que especifica la fracción de los datos de entrenamiento que se deben considerar como datos de validación.
- Número de épocas: las epochs determina el número de veces en las que todos los datos de entrenamiento han pasado por la red neuronal en el proceso de entrenamiento. Para el elegir el número de las epochs este se incrementa hasta que la métrica accuracy con los datos de validación empieza a decrecer, incluso cuando la accuracy de los datos de entrenamiento continúa incrementándose es cuando se detecta un potencial sobreajuste u overfitting.

La figura 36 muestra la configuración del proceso en código de compilación y entrenamiento de la nueva red.

Figura 36

Compilación y entrenamiento de la red.

```
resnet_model.compile(
        optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    resnet_model.fit(train_ds, validation_data=val_ds, epochs=7)
```

Para la configuración de los parámetros de números de epochs se genera un script en Google colab, para el entrenamiento y evaluación de este variando el número de épocas, así como la cantidad de imágenes, aprovechando el computo en la nube. Se debe generar una situación similar a la que se implementa en la Raspberry por lo que se realiza pruebas con un Data set conformado de imágenes captadas por la cámara del sistema, para el análisis se emplean dos categorías “Manzanas Rojas” y “Manzanas Verdes” con las características y resultados presentados en la tabla 11.

Tabla 11

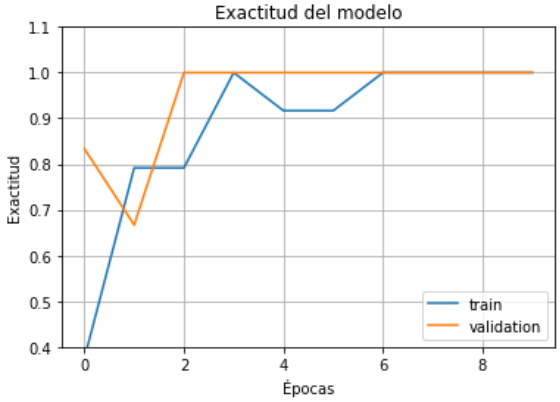
Variación de datos.

	Num. de imágenes	**Épocas**
Entrenamiento 1	60	10
Entrenamiento 2	60	7
Entrenamiento 3	50	10
Entrenamiento 4	50	7

Para el entrenamiento uno se obtuvieron las siguientes respuestas de validación. La figura 37 ilustra la gráfica del accuracy o Exactitud vs las epochs o Épocas, en la que para la época 6 el valor de exactitud deja de crecer pues llega a un cien por ciento, tanto en los datos de entrenamiento como en los de validación.

Figura 37

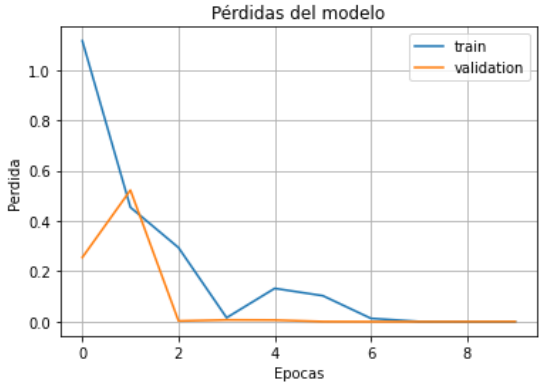
Exactitud del modelo del primer entrenamiento.



La figura 38 muestra el comportamiento de la función de pérdida durante el entrenamiento, de la misma manera se observa que para la época 6 el valor de perdida es prácticamente cero, tanto para los datos de entrenamiento como para los de validación.

Figura 38

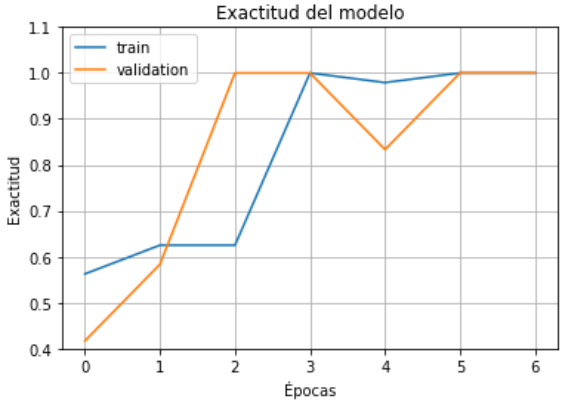
Pérdidas del modelo del primer entrenamiento.



Ya para el entrenamiento 2 se mantiene el tamaño del Data set y reduce a 7 el número de épocas de entrenamiento, obteniendo el resultado ilustrado por la figura 39, en la que se mantiene el valor de exactitud al cien por ciento a partir de la época 6.

Figura 39

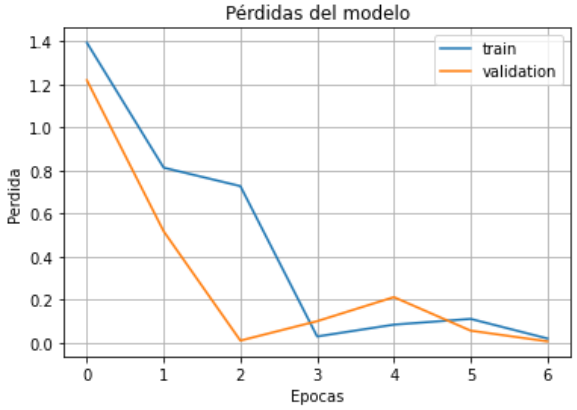
Exactitud del modelo del segundo entrenamiento.



La función de perdida para el entrenamiento 2 se comporta como lo muestra la figura 40 en la que las pérdidas tienden a cero desde la época 6, para los datos de entrenamiento y validación.

Figura 40

Pérdidas del modelo del segundo entrenamiento.

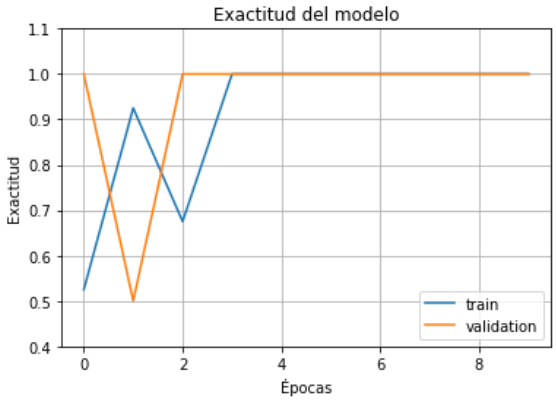


Para el entrenamiento 3 se reduce el número de imágenes que conforma el Data set a 50 pero se vuelve a incrementar las épocas de entrenamiento a 10, dando como resultado el comportamiento presentado en la figura 41 en el que ya para la época 4 la exactitud llega a su

valor máximo y se mantiene, sin embargo, existen picos y valles que no muestran un comportamiento de crecimiento regular del modelo.

Figura 41

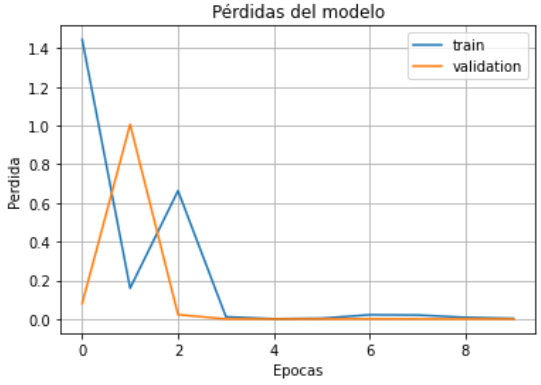
Exactitud del modelo del tercer entrenamiento.



Por su parte para el entrenamiento 3 la función de pérdida tiende a cero a partir de la época 4, y varia levemente en los 6 restantes, como se lo ilustra la figura 42.

Figura 42

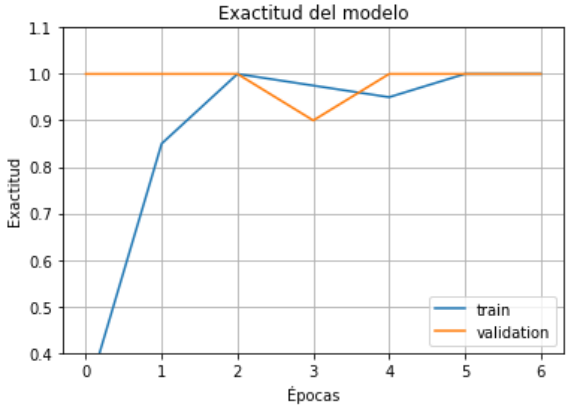
Pérdidas del modelo del tercer entrenamiento.



Finalmente, para el entrenamiento 4 se reducen los valores del Data set a 50 y el de número de épocas a 7 obteniendo el resultado que muestra la figura 43, en la que los valores de exactitud mantienen un crecimiento hasta la época 6 en la que llegan a su valor máximo y se mantienen.

Figura 43

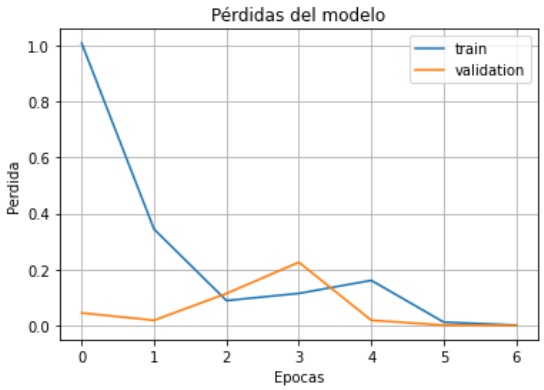
Exactitud del modelo del cuarto entrenamiento.



La función de pérdida por su parte para el entrenamiento 4 se comporta como la figura 44, en la que la pérdida tiende a cero a partir de la época de entrenamiento.

Figura 44

Pérdidas del modelo del cuarto entrenamiento.



Tras la realización de pruebas y evaluaciones del comportamiento del modelo variando el tamaño de datos y el número de épocas se puede afirmar que para el proyecto será conveniente trabajar con un número reducido de imágenes, para el caso serán 50, 25 por cada objeto, y 7 épocas de entrenamiento, pues con estos valores el resultado ha sido positivo y permiten reducir el consumo de recursos de procesamiento al momento de entrenar el modelo en el microordenador.

Una vez implementado el Servicio en la nube y el Servicio Local, la capa lógica está lista para poder ser añadida y empleada en la capa de aplicación en la sección del backend, dando así uso al consumo de estos dos servicios, que juntos permiten al sistema de clasificación cumplir con los objetivos propuestos para el proyecto.

Capa de aplicación

En este apartado se explica los códigos de programación que permiten que se ejecute las diferentes funciones ofrecidas al usuario en la interfaz, como se detalló en el capítulo anterior existen dos interfaces para el cliente, una interfaz local y una remota, la interfaz local posee las propiedades de supervisión, gestión y análisis, y la interfaz remota comprende únicamente con la supervisión y el análisis. Estas dos interfaces se basan en una estructura similar, abarcan con las capas Backend y Frontend, permitiendo mantener una correcta estructura, para una mejor comprensión se detallan en dos secciones, siendo la primera el Frontend y la siguiente el Backend.

Frontend

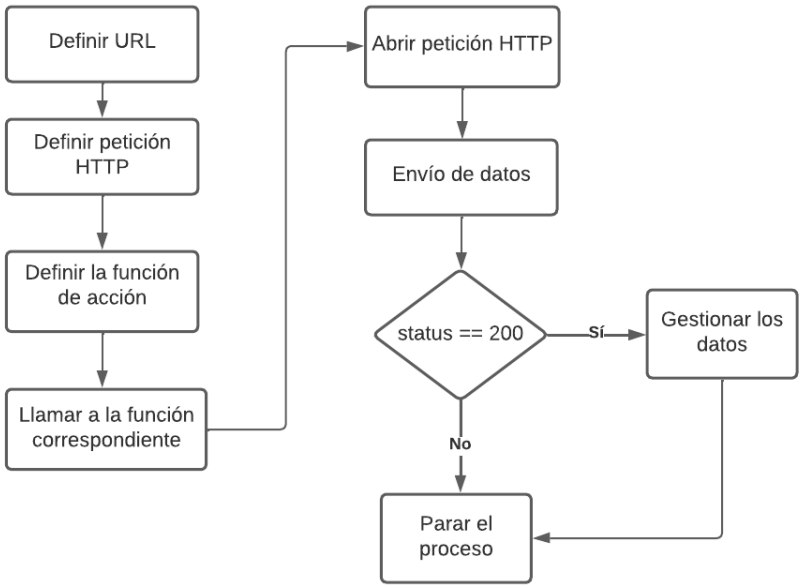
Como se explicó en el capítulo anterior, la aplicación debe contar con una estructura apropiada para un correcto funcionamiento, por lo que en esta parte se desarrolla la forma estética del programa, se utiliza una platilla de Dashboard desarrollada por Gentelella, esta se basa en HTML, CSS y JavaScript, con estos elementos se realizan apariencias llamativas para la aplicación, los controles serán especificados en la capa del Backend.

Dentro de esta capa se hace una importante división de los servicios al usuario, el primer servicio es el de la nube y el segundo es el local, el servicio en la nube usa como núcleo de predicción el servicio cognitivo de Azure, Image Analyzsis, el cual permite predecir el objeto al ingresar una imagen, y en el servicio local se maneja como núcleo de predicción la red neuronal ResNet-50.

En la figura 45 se detalla mediante un diagrama de flujo el funcionamiento que tiene esta capa, donde se muestra que el primer paso es definir la URL a dónde se debe conectar la aplicación, posteriormente se define la petición HTTP con la clase HttpRequest(), a continuación, se define la función de acción que ha solicitado el usuario, una vez definida la función se realiza la llamada, con esta información se procede a abrir la petición HTTP, en donde se termina de definir la dirección URL indicando si es un método POST o GET, a continuación, teniendo toda la información necesaria se envían los datos al servidor, y se procede a esperar una respuesta, si la respuesta de estado del servidor es de código 200, indica que se ha realizado con éxito la petición por lo que se procede a gestionar el pedido con los datos enviados, por el contrario si el estado recibido es diferente a 200 muestra que ha existido algún inconveniente en el proceso de envío por lo que no se puede realizar ninguna acción y se detiene el proceso.

Figura 45

Envío de información del Frontend.



Este funcionamiento de las peticiones HTTP del frontend son similares tanto para la interfaz local como la remota, sin embargo, cuentan con una interfaz distinta y por ende URLs diferentes que permiten la conexión con otras páginas web.

Interfaz Local. Como se lo ha detallado anteriormente esta interfaz cuenta con acceso directo a la Raspberry, por lo que es importante tener en cuenta esta característica, debido a que se puede alterar el funcionamiento del sistema.

Es necesario conocer las rutas a las cuales se dirigen los botones de la aplicación, de este modo en la tabla 12 se muestra todos los comandos existentes.

Tabla 12

Rutas y funciones.

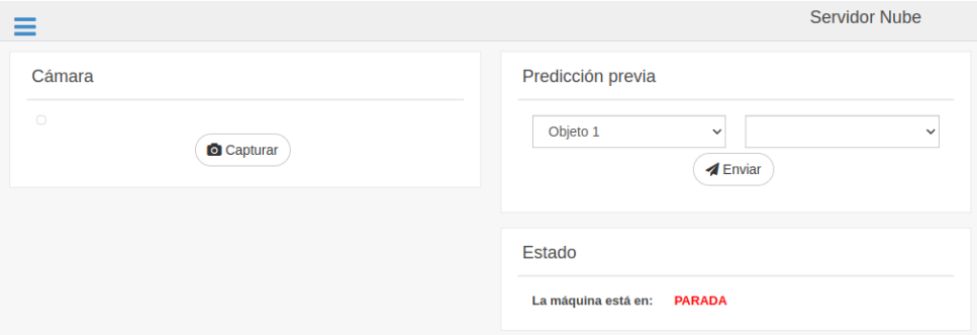
Controles	Ruta HTTP	Función POST
Servicio en la Nube		
Encender Stream	http://localhost:8088/Nube	startStreaming
Apagar Stream	http://localhost:8088/Nube	stopStreaing
Capturar	http://localhost:8088/Nube	CapturarAzure
Envío	http://localhost:8088/Nube	Envío
Inicio	http://localhost:8088/Nube	EmpezarProceso
Pausa	http://localhost:8088/Nube	PausaAzure
Parada	http://localhost:8088/Nube	PararAzure
Descargar	http://localhost:8088/Nube	Download
Servicio Local		
Encender Stream	http://localhost:8088/Local	startStreaming
Apagar Stream	http://localhost:8088/Local	stopStreaing
Capturar	http://localhost:8088/Local	Capturar

Controles	Ruta HTTP	Función POST
Entrenar	http://localhost:8088/Local	Entrenando
Eliminar	http://localhost:8088/Local	Borrar
Crear	http://localhost:8088/Local	Crear
Cargar	http://localhost:8088/Local	Cargar
Inicio	http://localhost:8088/Local	IniciarRs
Pausa	http://localhost:8088/Local	PausaRs
Parada	http://localhost:8088/Local	ParарRs
Descargar	http://localhost:8088/Local	download

Las interfaces se dividen en base al servicio local y servicio en la nube, en la figura 46 se muestra la parte inicial de la interfaz de servicio, en la parte izquierda se puede capturar la imagen y la lista con los objetos analizados se visualizan en el bloque de “Predicción previa”, en donde el usuario debe elegir el número y el objeto ingresado. En la parte inferior a ese bloque se presenta el estado del proceso, en la cual se logra visualizar la condición de los actuadores.

Figura 46

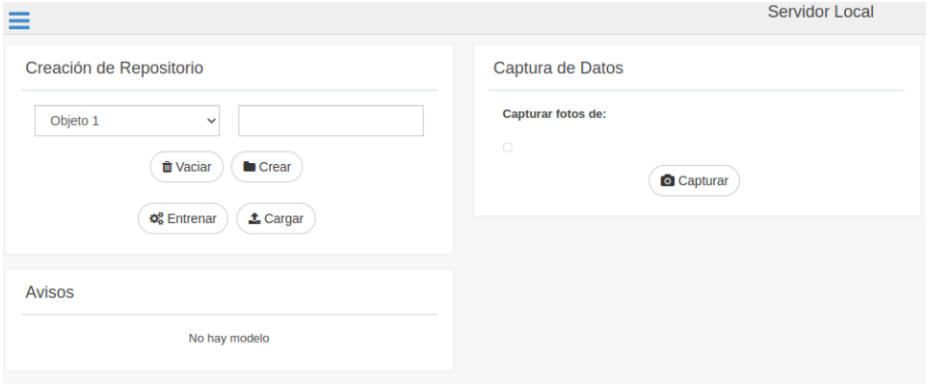
Bloque inicial de la página.



En cuanto al servicio local como se ilustra en la figura 47, al inicio de su interfaz se puede observar tres apartados, en el bloque de “Creación de Repositorio” se ingresa el nombre del objeto a clasificar, también hay las opciones de entrenar, cargar o vaciar, el uso de cada uno va a depender según los datos que se dispongan en el momento, y se informa en el apartado de “Avisos”, finalmente en el bloque de “Captura de Datos” sirve para tomar fotos de los objetos a entrenar para posteriormente ser clasificados en el sistema.

Figura 47

Bloque inicial de la página local.



Tanto la interfaz del servicio en la nube como el local, se puede observar el panel de control del sistema, junto con la información de los objetos a clasificar, como se muestra en la figura 48.

Figura 48

Bloque de control.

Posterior al panel de control se tiene el bloque de video en tiempo real del proceso y una gráfica de dona que muestra la confiabilidad de cada predicción junto con los contadores de los objetos clasificados, como se lo muestra en la figura 49.

Figura 49

Bloque de vídeo y resultados.

Finalmente, en la parte inferior de las páginas se puede observar dos graficas que indican el porcentaje de confianza frente a cada contador, como se lo indica en la figura 50, mediante esta gráfica se puede realizar diversos análisis de funcionamiento tanto del sistema como de la red usada.

Figura 50

Bloque de análisis.

Para usar la aplicación se debe primero ejecutar el servidor de Django y posteriormente entrar a cualquier navegador e ingresar al servidor local.

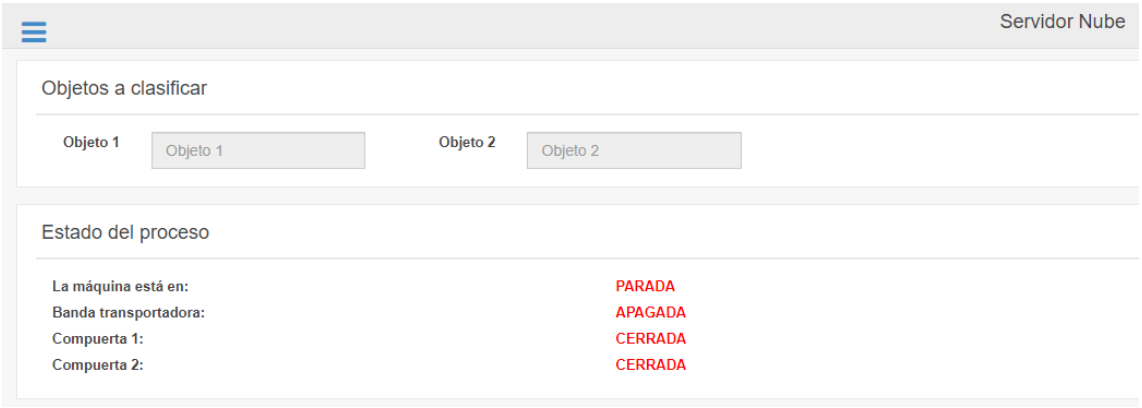
Interfaz Remota. Esta interfaz tiene acceso a la supervisión y el análisis de los datos del sistema, a diferencia de la interfaz local en esta no se podrá realizar un control directo a la Raspberry, esto es debido a que esta interfaz es pública y se pude acceder desde una URL especifica desde cualquier dispositivo con conexión a internet.

Al igual que en la interfaz local, en esta interfaz también se realiza una división para una mejor experiencia de supervisión para el usuario, la primera parte muestra el diseño de una interfaz del servicio en la Nube y la segunda un servicio local, sin embargo, el diseño de las dos es similar.

Dentro de la interfaz de servicio en la Nube como en la Local se comienza mostrando los objetos a clasificar y posteriormente, el estado del proceso, en el cual muestra la condición de los actuadores del sistema en este momento, como se lo puede observar en la figura 51.

Figura 51

Bloque de inicio de página remota.

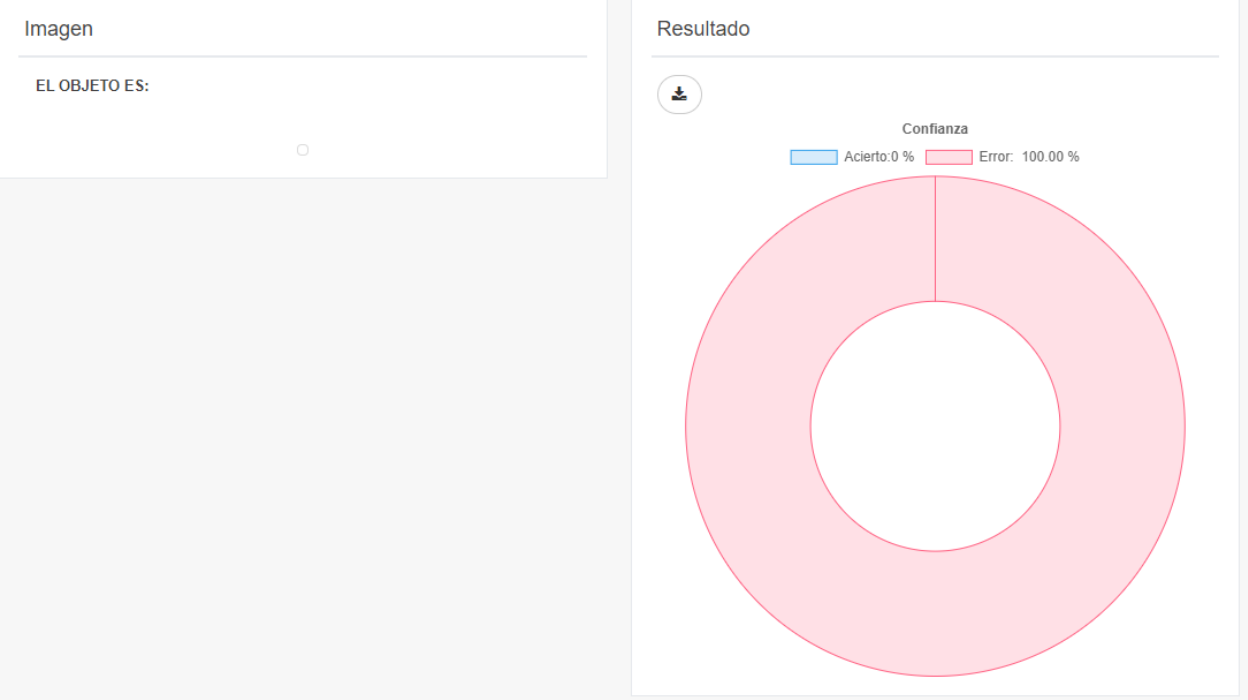


A continuación, como se indica en la figura 52, se ilustra la imagen captada en el sistema junto con su predicción y al lado derecho un diagrama de dona el cual refleja el

porcentaje de acierto y su error, mediante esta grafica se puede asimilar que tan precisa es la red del servicio en la Nube, Azure, o Local, ResNet-50, y realizar un análisis.

Figura 52

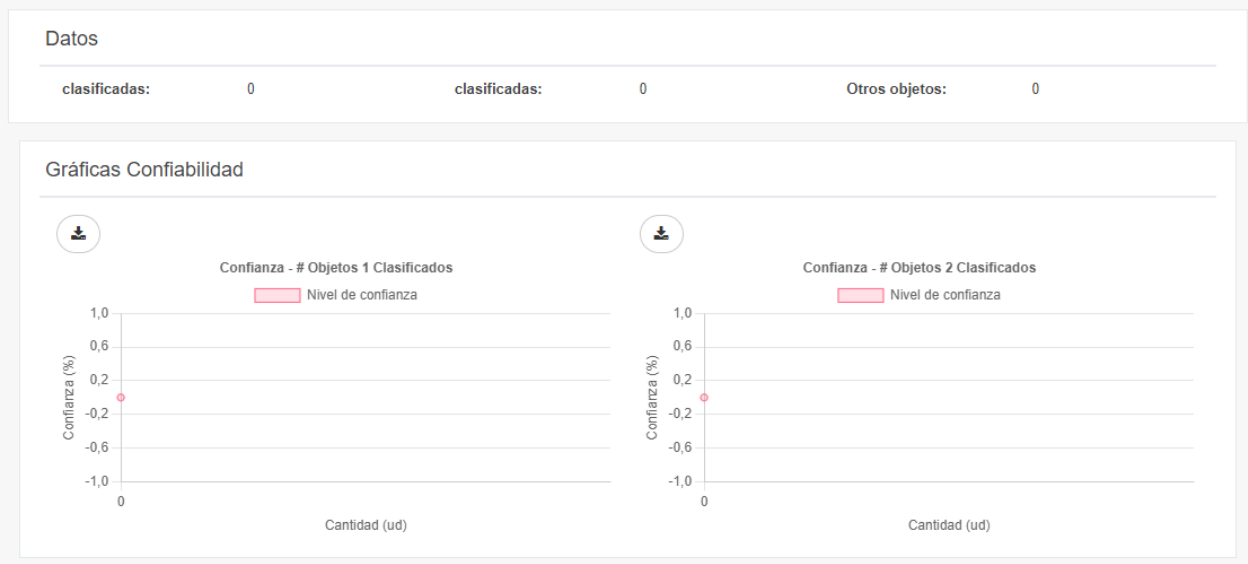
Bloque de vídeo y resultados de página remota.



En la figura 53 se denota en primer lugar los datos de los contadores en tiempo real, y a continuación las gráficas de confianza vs cantidad, estás gráficas son muy importantes para realizar un análisis del desempeño de la red en la Nube o Local, muestra un histórico de cuan efectiva es la predicción en cada objeto detectado y clasificado, por lo que en base a estas gráficas se puede obtener diversas conclusiones.

Figura 53

Bloque de contadores y análisis de página remota.



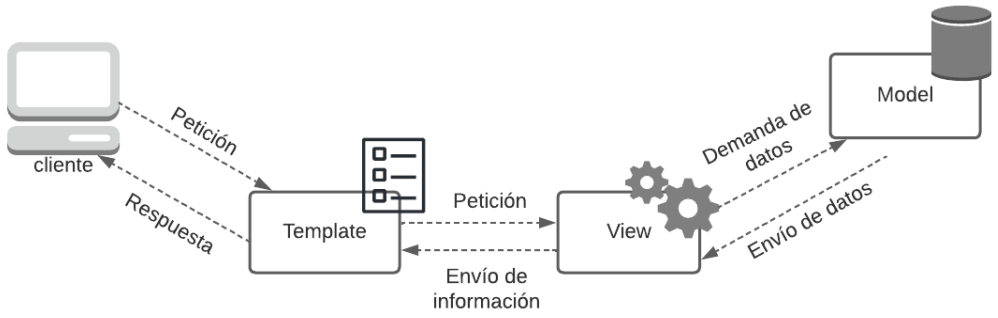
Backend

Dentro de esta capa se desarrollan las funciones que permiten realizar las diferentes acciones que solicita el usuario, por lo que se considera el núcleo del proyecto.

Django. Este framework cuenta con una estructura especifica la cual se la puede observar en la figura 54, esta estructura empieza por un cliente, el cual mediante el uso de un navegador web realiza peticiones a un servidor, los templates es la parte visual que se presenta al usuario, en la cual se puede interactuar, en cuanto al view es el encargado de gestionar las peticiones del usuario que son enviadas entre los templates y el model, y finalmente el model es la parte en la cual se gestiona los datos, se almacena y se envía la información solicitada por el usuario.

Figura 54

Estructura de Django.



Al realizar esta estructura se generan ventajas como: funcional, mantenible y escalable, por lo que se garantiza una correcta funcionalidad.

Creación de aplicación

El primer paso es la instalación del framework Django, la sintaxis del comando para la instalación de este se lo muestra a continuación.

pip install Django

Interfaz Local

Posteriormente a la instalación se debe dirigir a la ubicación en la cual se va a trabajar en el proyecto mediante comandos *cd* a través de la consola de Windows para la creación del proyecto. Para tener una estructura organizada se debe crear aplicaciones las cuales se conecten con el proyecto principal de Django, estas aplicaciones permiten ser reutilizadas, para el presente proyecto es necesario la creación de dos aplicaciones, la cual una es llamada servicioLocal y la siguiente servicioNube y son conectadas directamente al proyecto principal llamado Clasificador, en la figura 55 se puede observar el árbol del proyecto con sus respectivas aplicaciones.

Figura 55

Árbol de proyecto.

En la figura 56 se ilustra el contenido del proyecto principal los cuales cuentan con los siguientes ficheros: __init__.py, asgy.py, setting.py, urls.py, wsgi.py y esencialmente se encuentra el archivo manage.py, este permite la creación de aplicaciones, la ejecución del servidor y más, es un elemento importante dentro del proyecto, los archivos creados automáticamente cuentan con una funcionalidad importante por lo que no es recomendable eliminarlos.

Figura 56

Archivos principales.

Para la unión de las aplicaciones con el proyecto principal se debe especificar el nombre de estas dentro del fichero “setting.py” en el apartado INSTALLED_APPS, como lo muestra la figura 57.

Figura 57

Unión de aplicaciones.

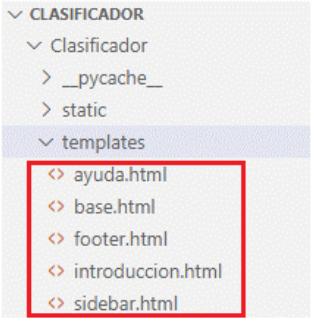
```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'channels',
    'servicioNube',
    'servicioLocal',
]
```

Templates

En una aplicación de Django los templates son los archivos que se desarrollan con HTML, CSS y JavaScript, estos son las interfaces en las cuales el usuario podrá interactuar y realizar las respectivas peticiones al servidor, en la figura 58 se muestra los templates usados para la base del proyecto.

Figura 58

Archivos HTML.



- **Ayuda.html:** esta ventana brinda información sobre cómo usar la interfaz tanto la de servicio local como la del servicio en la nube. En esta ventana el usuario únicamente se informará sobre la página mas no se mostrará botones de control.
- **Base.html:** este archivo contiene el formato general, aquí se encuentran las librerías y diseños a usar en cada página.
- **Footer.html:** este archivo es el diseño del pie de página que se muestra al final de cada página web.
- **Introducción.htm:** en esta página se encuentra una descripción del proyecto clasificador, de los servicios en la nube y el local, esta también es únicamente informativa para el usuario, desde esta interfaz no puede controlar ninguna parte del sistema.
- **Siderbar.html:** este archivo permite la navegación entre las interfaces explicadas.

URL/View

En este apartado se encuentran los enlaces para la respectiva conexión entre templates, en la figura 59 se ilustran los enlaces para conectarse a las páginas de ayuda, introducción y también a las URLs de las aplicaciones de servicio en la nube y servicio local.

Figura 59

URLs del proyecto principal.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.introduccion,
name="introduccion"),
    path('ayuda/', views.ayuda, name="ayuda"),
    path('Local/', include('servicioLocal.urls')),
    path('Nube/', include('servicioNube.urls')),
]
```

Model

En este apartado se gestionan las peticiones realizadas por el usuario, donde indica que acción debe tomar, como se muestra en la figura 60, cuando el usuario requiera dirigirse a la página de introducción se realiza el respectivo enlace y se renderiza el template requerido, lo mismo sucede cuando se desea ir a la página de ayuda.

Figura 60

Modelos del proyecto principal.

```
def introduccion(request):
    return render(request, "introduccion.html")

def ayuda(request):
    return render(request, "ayuda.html")
```

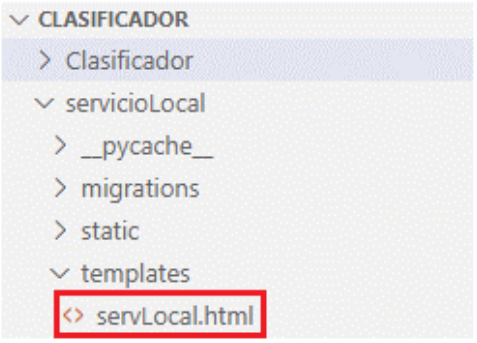
Servicio Local

Templates

Estos archivos son desarrollan con HTML, CSS y JavaScript, en esta página el usuario podrá realizar peticiones de control al sistema, el núcleo de este servicio es una CNN, ResNet-50, en la figura 61 se puede observar el archivo HTML usado para esta página.

Figura 61

Archivo HTML del servicio local.



- **servLocal.html:** en esta ventana se realizan todas las peticiones de funcionamiento para la CNN, ResNet-50, y se puede interactuar con el sistema, se requiere de un solo archivo debido a que hereda las características del template base.

URL/View

En la figura 62 se ilustra el enlace al cual se conecta para hacer el llamado a la página del servicio local, estos enlaces deben ser únicos. Para la definición de una URL se debe hace mediante la librería “path”, y esta se debe encontrar dentro de “urlpatterns”, la cual es una variable reservada por Django, y le permite entender al framework que dentro de esta variable se encuentran los enlaces de las páginas.

Figura 62

URL de la aplicación local.

```
urlpatterns = [
    path('', views.Local, name="Local"),
]
```

Model

Las funciones para gestionar dentro del servicio local son las que el usuario puede realizar desde la interfaz, estás se activan según la petición POST que se genere desde el Frontend, mediante este proceso se evita la creación de diversas URLs para la llamada a cada función, en la figura 63 se puede mostrar las funciones usadas.

Figura 63

Modelos del servicio local.

```
if request.method == "POST":
    if "btnBorrar" in request.POST: # Borra los datos···
    elif "btnTags" in request.POST: # Crea las carpetas···
    elif "btnCargar" in request.POST: # Carga los datos···
    elif "btnCapEn" in request.POST:  # -Capturar fotos de entrenamiento···
    elif "btnIniciarRs" in request.POST: # Inicia el proceso···
    elif "btnPararRs" in request.POST: # Detiene el proceso···
    elif "btnEntrenar" in request.POST: # Entrena con los datos···
    elif "btnPausaRs" in request.POST: # Pausa el proceso···
return render(request, "servLocal.html", contexto)
```

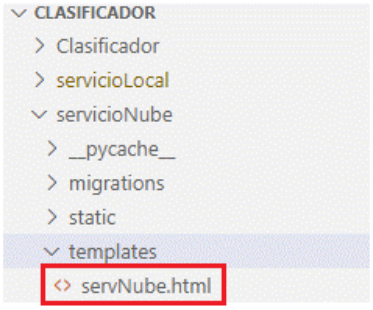
Servicio en la Nube

Templates

Estos templates también son diseñados con HTML, CSS y JavaScript, en esta ventana se tiene como red principal el servicio cognitivo, Computer Vision de Azure, al igual que en el servicio local este hereda los diseños del template base, por lo que es necesario solo un archivo como lo muestra la figura 64.

Figura 64

Archivo HTML del servicio en la nube.



URL/View

En esta etapa se define la dirección de enlace a la cual se debe conectar cuando el usuario desee ir a la página de servicio en la nube, por lo que se debe usar la librería “path”, y detallar dentro de la variable “urlpatterns”, como lo indica la figura 65.

Figura 65

URL de la aplicación del servicio en la nube.

```
urlpatterns = [
    path('', views.Nube, name="Nube"),
]
```

Model

Mediante las funciones que se muestran en la figura 66 se pueden realizar distintas acciones según las peticiones pedidas por el usuario.

Figura 66

Modelos del servicio en la nube.

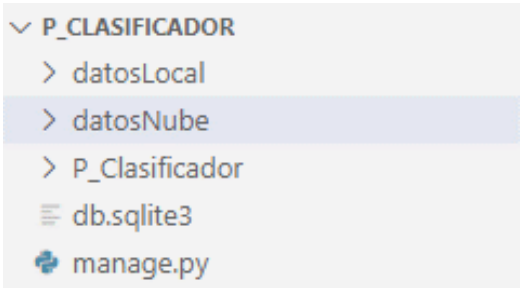
```
if request.method == "POST":
    if "btnObjR" in request.POST: #Obtiene los datos···
    elif "btnCapAz" in request.POST: # Captura la imagen···
    elif "btnIniciarAz" in request.POST: # Inicia el proceso···
    elif "btnPausaAz" in request.POST:# Pausa el proceso···
    elif "btnPararAz" in request.POST:# Para el proceso···
return render(request,"servNube.html", contexto)
```

Interfaz Remota

En la figura 67 se muestra el árbol del proyecto para la creación de la interfaz remota, en el desarrollo de esta interfaz se podrá notar que es muy similar a la interfaz local, cuenta también con las aplicaciones de los datos locales y en la nube.

Figura 67

Árbol del proyecto de interfaz remota.



Para poder enlazar las aplicaciones al proyecto principal se debe dirigirse al archivo de configuraciones, y añadir los nombres de las aplicaciones como se lo muestra en la figura 68.

Figura 68

Unión de aplicaciones de interfaz remota.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'channels',
    'datosLocal',
    'datosNube',
]
```

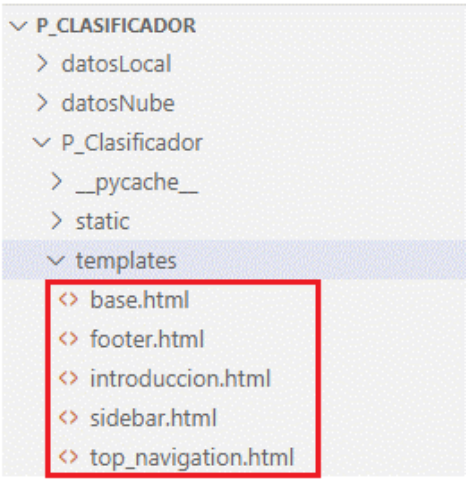
En cuanto a la estructura del programa es similar a la interfaz local, cuenta con templates, views y models, en el proyecto base se encuentra los templates que tienen los diseños bases para las páginas web.

Template

En esta capa se encuentran los diseños HTML para la visualización de las páginas web a los usuarios, como se lo muestra en la figura 69.

Figura 69

Archivos HTML para la interfaz remota.



- **base.html:** este archivo contiene los diseños bases para que puedan heredar el resto de páginas web, de esta manera se puede mantener un diseño uniforme para todas las interfaces.
- **Footer.html:** este archivo contiene el estilo del pie de página que se puede observar al final de cada página.
- **Introducción.html:** dentro de esta ventana se muestra una descripción sobre el proyecto junto con los servicios anexados: servicio local y servicio en la nube.
- **Sidebar.html:** esta ventana permite la navegación entre las diversas páginas web existentes.
- **Top_navigation.html:** este archive contiene el diseño del encabezado de cada página.

URL/View

En cuanto a las URLs, se deben declarar en el archivo “urls.py” de la misma manera que en la interfaz local se usa la librería “path” y la variable “urlpatterns”, al ser el proyecto base también se debe enlazar las aplicaciones usadas para poder tener una comunicación entre todas las páginas, esto se lo puede observar en la figura 70.

Figura 70

URLs de proyecto de interfaz remota.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.introduccion, name="introduccion"),
    path('Local/', include('datosLocal.urls')),
    path('Nube/', include('datosNube.urls')),

] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Model

Para este apartado al ser el proyecto base, solo se cuenta con la renderización de la página web introducción, mediante las líneas de código que se muestra en la figura 71, se procede a renderizar el HTML deseado.

Figura 71

Modelo de interfaz remota.

```
def introduccion(request):
    return render(request, "introduccion.html")
```

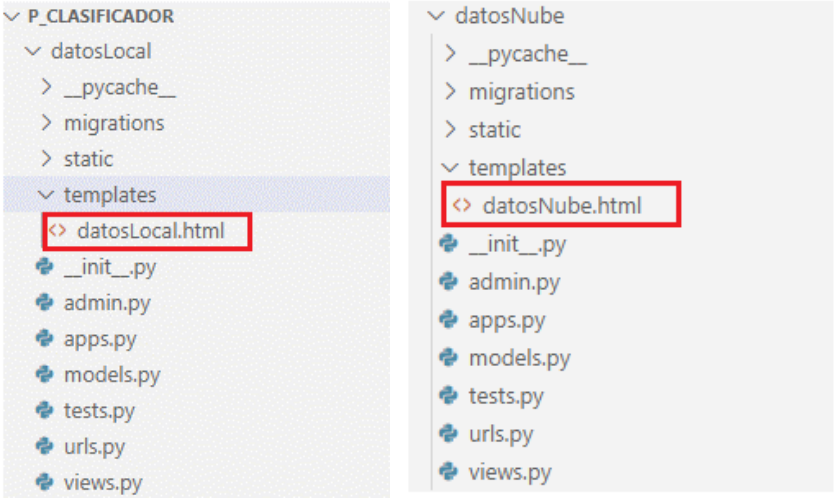
Las aplicaciones de datosLocal y datosNube cuentan con la misma división que la del proyecto general, por lo que, tienen archivos similares las dos aplicaciones.

Template

Como se puede observar en la figura 72, la aplicación *datosLocal* y *datosNube* usan un solo archivo HTML, esto debido a que heredan los diseños del template base.

Figura 72

Archivos HTML para servicios en la nube y local.



URL/View

En cuanto a los enlaces para conectarse con cada una de las interfaces, se los debe especificar en el archivo “urls.py” de cada aplicación, en la figura 73 se puede observar la creación del enlace para la página de *datosLocal* y en la figura 74 se ilustra la creación para la conexión con los *datosNube*.

Figura 73

URL para servicio en la local.

```
urlpatterns = [
    path('', views.Local, name="Local"),
]
```

Figura 74

URL para servicio en la nube.

```
urlpatterns = [
    path('', views.Nube, name="Nube"),
]
```

Model

Para poder mostrar los datos al usuario se debe captar la información enviada por JSON desde la Raspberry, es importante tener en cuenta los nombres de los valores de los atributos del diccionario, mediante estos nombres se pueden enlazar a la interfaz remota y mostrar los datos necesarios al usuario, los nombres van a variar según el servicio requerido, en la figura 75 se observan los datos del servicio Local, y en la figura 76 se ilustra los datos para el servicio en la nube.

Figura 75

Modelo del servicio local.

```
@csrf_exempt
def Local(request):
    global obj1Local, obj2Local, cont1Local, cont2Local, confLocal, conf1Local,
conf2Local, predLocal
    global estado, foto, servoM1, servoM2, banda
    if request.method == "POST":
        obj2Local = request.POST['obj2Local']
        obj1Local = request.POST['obj1Local']
        cont1Local = request.POST['cont1Local']
        cont2Local = request.POST['cont2Local']
        confLocal = request.POST['confLocal']
        conf1Local = request.POST['conf1Local']
        conf2Local = request.POST['conf2Local']
        predLocal = request.POST['predLocal']
        estado = request.POST['estado']
        servoM1 = request.POST['servoM1']
        servoM2 = request.POST['servoM2']
        banda = request.POST['banda']
        foto = request.POST['foto']
    return render(request, "datosLocal.html")
```

Figura 76

Modelo del servicio en la nube.

```
@csrf_exempt
def Nube(request):
    global obj1Nube, obj2Nube, cont1Nube, cont2Nube, cont3Nube, confNube,
conf1Nube, conf2Nube
    global predNube, estado, foto, servoM1, servoM2, banda
    if request.method == "POST":
        obj2Nube = request.POST['obj2Nube']
        obj1Nube = request.POST['obj1Nube']
        cont1Nube = request.POST['cont1Nube']
        cont2Nube = request.POST['cont2Nube']
        cont3Nube = request.POST['cont3Nube']
        confNube = request.POST['confNube']
        conf1Nube = request.POST['conf1Nube']
        conf2Nube = request.POST['conf2Nube']
```

```
        predNube = request.POST['predNube']
          estado = request.POST['estado']
          servoM1 = request.POST['servoM1']
          servoM2 = request.POST['servoM2']
          banda = request.POST['banda']
          foto = request.POST['foto']
    return render(request, "datosNube.html")
```

Predicción de elementos. El apartado de predicción es esencial para el desarrollo del proyecto, como se lo ha ido mencionando en el transcurso de este, se cuenta con dos servicios claves para el funcionamiento del sistema, el servicio en la nube que se encuentra potenciado por un servicio cognitivo de Azure llamado computer visión, y de este se utiliza la función tag_image, y el servicio local se encuentra desarrollado en base a una CNN, que es la ResNet-50, esta red convolucional permite la predicción de objetos mediante el uso de imágenes.

El uso de la característica de predicción es exclusivo para la interfaz Local, debido a que tiene una conexión directa con el sistema y los controladores, en la interfaz remota únicamente tiene la característica de poder observar los datos que son enviados a través de la Raspberry, por lo que no es necesario que se ejecute estas redes neuronales en la interfaz remota.

Interfaz Local

Como se lo explicó anteriormente, el uso de la predicción se lo implementa solo en la interfaz local por la conexión directa que tiene a los actuadores y a todo el programa en general, y esto puede causar daños, una vez que se procese esa información se lo envía a la interfaz en la nube para mostrarlo al usuario.

Azure

Dentro del servicio en la Nube se tiene el servicio cognitivo Azure, del cual se utiliza la función tag_image, para poder usar esta función se debe primero importar dos librerías importantes las cuales se pude observar en la figura 77. La primera librería permite conectarse

al usuario al servicio ComputerVisionClient, el cual permite poder acceder al servicio cognitivo que se desea, y la segunda librería sirve para poder validar las credenciales del servicio cognitivo que se va a utilizar, como se lo había explicado anteriormente estos servicios son de pago, cuando se usa la prueba gratuita o se cancela un plan la plataforma le asigna una contraseña para el uso de estos servicios.

Figura 77

Librerías de computer visión.

```
from azure.cognitiveservices.vision.computervision import ComputerVisionClient
from msrest.authentication import CognitiveServicesCredentials
```

Una vez llamadas las librerías se procede a colocar sus credenciales y el endpoint, con este valor permite saber a qué servicio se debe dirigir, en la figura 78 se pude observar cómo se ingresa estos datos.

Figura 78

Configuración de credenciales del servicio.

```
KEY = 'Coloque su contraseña en esta variable'
ENDPOINT = 'https://tesis-instv2.cognitiveservices.azure.com/'
```

Posteriormente se debe ingresar las credenciales previamente colocadas, para esto se declara una nueva variable y se hace uso de la función ComputerVisionClient de la librería que anteriormente llamamos, dentro de esta función se ingresan los datos para poder autentificarse e indicar el servicio al cual debe enlazarse, en la figura 79 se ilustra lo explicado.

Figura 79

Asignación de datos.

```
cv_client = ComputerVisionClient(ENDPOINT, CognitiveServicesCredentials(KEY))
```

A continuación, se realiza la función de predicción, en la cual se lleva a cabo el proceso de identificación de objetos, por lo que antes de realizar la predicción se debe abrir el puerto de

la cámara, seguidamente se captura la imagen, se la guarda en una dirección en específico y se cierra el puerto de la cámara, como lo ilustra la figura 80.

Figura 80

Función de acción de cámara.

```
camera = cv2.VideoCapture(idCam)
    if not camera.isOpened():
        idCam = 0
        print(f"Cambio de puerto en cámara a {idCam}")
        camera = cv2.VideoCapture(idCam)
    _, img = camera.read()
    cv2.imwrite(direccionF, img)
    camera.release()
```

Posterior al proceso de capturar la imagen, se abre la misma en modo lectura, y se hace el llamado de la API de Azure, en la figura 81 se lo ilustra con el nombre cv_client, a continuación, se indica la función que se desea usar, en este caso es tag_image, esta función analizará la imagen que le ingresada y entregará una serie de datos, dentro de ellos se encuentran las etiquetas de los objetos encontrados en la imagen, también el porcentaje de probabilidad de cada uno, estos dos valores son muy importantes para el proyecto por lo que se los almacena en un vector y al finalizar la función se los devuelve en diferentes variables.

Figura 81

Función de llamada de API.

```
with open(direccionF, mode='rb') as image_stream:
        # Llamar a la API Azure
        tags_result_remote = cv_client.tag_image_in_stream(image_stream, language="es")
        # Obtención de predicción de Azure y confianza
        if (len(tags_result_remote.tags) == 0):
            aviso = "No hay objetos detectado."
        else:
            for tag in tags_result_remote.tags:
                tagsOb.append(tag.name.capitalize())
                tagConf.append(round(tag.confidence*100,2))
                global listaOb
                listaOb = tagsOb
    return tagsOb,tagConf
```

Al igual que Azure, esta red únicamente funciona en la interfaz local, debido a que tiene conexión directa con los actuadores del sistema, y también con los controles de entrenamiento, y estas características no tiene la interfaz web. Esta red se utiliza cuando el servicio en la nube no identifica correctamente el objeto que se desea clasificar, en ese caso se procede a usar este servicio, en el cual se debe seguir un sencillo proceso para poder entrenar a la red con los objetos que se desea.

Para el uso de esta red primero se debe importar librería de tensorflow y el archivo resnet50 como se ilustra en la figura 82, el cual se explicó con detalle anteriormente, al llamar al archivo resnet50 automáticamente se importan todas las librerías necesarias para el funcionamiento de esta.

Figura 82

Librerías para uso de Resnet50.

```
from servicioLocal import resnet50
import tensorflow as tf
```

Antes de usar esta red, primero se debe realizar una base de datos en base a los objetos que se necesitan clasificar, por lo que el usuario primero deberá crear carpetas con los nombres de los objetos, el nombre de la carpeta es importante debido a que al momento de entrenar tomará este nombre como clase, y cuando se realice la predicción mostrará el nombre con el que se entrenó.

En la figura 83 se puede observar que el proceso para crear una carpeta, una vez que el usuario ingrese el nombre del objeto y presione el botón de envío, se activa la función para crear una carpeta con el mismo nombre del objeto, primero se valida que el texto ingresado sea correcto, posteriormente mediante el atributo “request”, se captura los datos desde el HTML, por lo que se obtiene el nombre enviado por el usuario. A continuación, se verifica que no

exista una carpeta con el mismo nombre, de ser así se informa al usuario que ya existe una carpeta con el nombre ingresado, caso contrario si es un nuevo nombre se crea la carpeta.

Figura 83

Función de creación de carpetas.

```
def carpetasTag(request):
    global aviso,bandCarpeta
    aviso = ""
    contexto = {"aviso": aviso}
    Eform = dato_Tag(request.POST)
    if Eform.is_valid():
        fileCarp = request.POST.get('txtTag')
        fileCarp = fileCarp.capitalize()
        DatosEn = DatosEntr(request)
        contexto.update(DatosEn)
        if(os.path.exists(dirRaspDatos+fileCarp)):
            aviso = "La carpeta ya existe."
            bandCarpeta = False
            contexto.update({"aviso": aviso})
        else:
            os.mkdir(dirRaspDatos+fileCarp)
            aviso = f"Se creó la carpeta {fileCarp}, por favor capture datos."
    else:
        aviso = "Error al ingresar los datos."
        bandCarpeta = False
        contexto.update({"aviso": aviso})
    return contexto
```

Una vez creada la carpeta, el usuario debe capturar fotos del objeto deseado, al realizar dicha acción se ejecuta el código que se observa en la figura 84, en el cual indica que primero activa el puerto de la cámara, de no existir ningún error, mantiene el puerto y captura la imagen en ese momento y se almacenan los datos en una variable, y a continuación, se cierra la cámara. Antes de guardar la imagen se debe analizar la ubicación y el nombre, por lo que primero se analiza el valor del selector en el que el usuario lo indicó, posteriormente se designa

a que carpeta ir y el nombre del archivo, para finalmente guardar la imagen en el lugar de la base de datos para el entrenamiento.

Figura 84

Función de captura de datos.

```
elif "btnCapEn" in request.POST:  # -Capturar fotos de entrenamiento
            global bandFr,idCam
            bandFr=True
            camera = cv2.VideoCapture(idCam)
            if not camera.isOpened():
                print("Cambio de puerto en cámara")
                idCam = 0
                camera = cv2.VideoCapture(idCam)
            _, img = camera.read()
            camera.release()
            if (SelecRes == "Objeto 1"):
                global contImOb1,contImOb2,dirFotoRes
                contImOb1 = contImOb1+1
                dirDatos = f"{dirRaspDatos}{Obj1ReN}/{Obj1ReN}{contImOb1}.jpg"
                dirFotoRes=f"/static/Datos/{Obj1ReN}/{Obj1ReN}{contImOb1}.jpg"
                contImOb2 = 0
                cv2.imwrite(dirDatos, img)
            else:
                contImOb2 = contImOb2+1
                dirDatos = f"{dirRaspDatos}{Obj2ReN}/{Obj2ReN}{contImOb2}.jpg"
                dirFotoRes=f"/static/Datos/{Obj2ReN}/{Obj2ReN}{contImOb2}.jpg"
                cv2.imwrite(dirDatos, img)
                contImOb1 = 0
            listaDatos = os.listdir(dirCarpDatos)
```

Luego de haber tomado la cantidad necesaria de fotos de cada objeto para el entrenamiento, se procede a entrenar la ResNet-50 con los datos almacenados, en la figura 85 se muestra las validaciones para evitar errores en el proceso del entrenamiento y la también la ejecución del entrenamiento.

De ser el caso que todo está correcto, se procede a entrenar la red llamando a la función “Entrenar” la cual se encuentra en el archivo “resnet50”, al ejecutar el código de

entrenamiento se demorará en finalizar dicha acción, el tiempo de demora dependerá del número de imágenes y épocas que se haya ingresado a la red. Una vez entrenado se procede a guardar el modelo y a cargarlo en el sistema para poder dar uso en la identificación de objetos.

Figura 85

Validación de datos.

```
if (len(listArchivos)<2):
        aviso="No hay datos suficientes para entrenamiento"
        bandEntren = False
    elif(os.path.exists(dirCargModelo)):
        aviso="Ya existe un modelo, presione cargar"
        bandEntren = False
    elif(len(listArchivos)==2):
        Obj1Rs=listArchivos[0]
        Obj2Rs=listArchivos[1]
        dirDatosObj1 = os.path.join(dirRaspDatos,Obj1Rs)
        listaDatos1=os.listdir(dirDatosObj1)
        dirDatosObj2 = os.path.join(dirRaspDatos,Obj2Rs)
        listaDatos2=os.listdir(dirDatosObj2)
        if(len(listaDatos1) >= numImagenes and len(listaDatos2) >= numImagenes):
            bandDatosModel = False
            bandDatos = False
            bandCargar = False
            bandEntren=True
            Obj1ReN=listArchivos[0]
            Obj2ReN=listArchivos[1]
            aviso = "Entrenando, espere por favor"
            bandModelo=False
            resnet50.Entrenar()
            global modelo_final, class_name
            modelo_final, class_name = CargarModelo(dirCargModelo, dirModelo)
            bandEntren=False
            aviso="Modelo listo para usar"
```

Después de haber entrenado y guardado el modelo, se puede dar uso en el sistema, para realizar predicciones con el modelo creado, en la figura 86 se ilustro el código para la

predicción en base a este método, al igual que en el servicio en la nube se debe primero activar la cámara, capturar la imagen, y guardarla en una dirección en específica. Para obtener una predicción en base al modelo se debe primero redimensionar las imágenes a la entrada de la red, con la función “predic” hace llamado a la función de predicción interna, y como parámetro se envía la imagen, la respuesta de esta función son valores flotantes los cuales representan la probabilidad de cada objeto, finalmente se elige el dato con mayor probabilidad junto con su etiqueta, para posteriormente mostrar al usuario la predicción que se realizó con el modelo basado en ResNet-50.

Figura 86

Función de predicción con Resnet-50.

```
img_height, img_width = 180, 180
    camera = cv2.VideoCapture(idCam)
    if not camera.isOpened():
        print("Cambio de puerto en cámara")
        idCam = 0
        camera = cv2.VideoCapture(idCam)
    _, img = camera.read()
    dirFotoRes2=""
    dirFotoRes2=guardarFotoR()
    cv2.imwrite(direccionF, img)
    camera.release()
    image = cv2.imread(direccionF)
    image_resized = cv2.resize(image, (img_height, img_width))
    image = np.expand_dims(image_resized, axis=0)
    pred = modelo_final.predict(image)
    valPredic=np.amax(np.multiply(pred,100))
    valPredic=np.float64(valPredic).item()
    valPredic=round (valPredic,2)
    prediccionR = class_name[np.argmax(pred)]
```

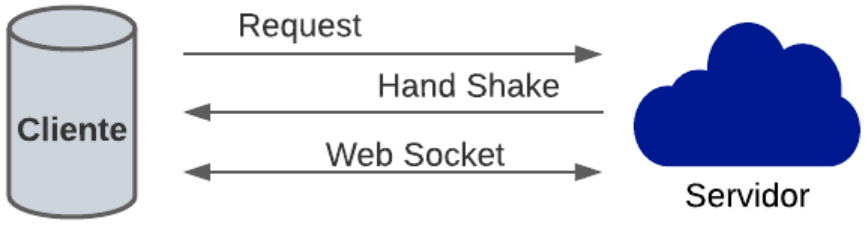
Comunicación de datos. Para la comunicación entre el cliente y el servidor, el uso del protocolo websocket resulta ser esencial, al crear un canal de comunicación dual permite mejorar la transmisión de datos.

Websocket

Como se puede observar en la figura 87 la conexión entre el servidor y el cliente, en donde el cliente mediante algún navegador web realiza una petición o request, esta acción permite el desarrollo de un Hand Shake entre el cliente y el servidor, una vez activada la conexión entre ellos, el intercambio de datos se lo realiza de modo bidireccional, la comunicación se lleva a cabo utilizando el mismo canal de conexión hasta que se finalice, esto sucederá cuando el servidor o el cliente termine la comunicación.

Figura 87

Diagrama de conexión de Web Socket.



Este protocolo se lo ha implementado tanto en la interfaz local como en la remota, de esta manera se asegura una comunicación full duplex.

Interfaz Local

Para poder hacer uso de este protocolo se debe instalar las librerías channels y channels-redis, esto se lo hace mediante los siguientes comandos:

pip install channels

pip install channels-redis

Posterior a la instalación, se requiere dirigirse al archivo de configuraciones “settings.py” y poder agregar como una aplicación más, como se lo muestra en la figura 88.

Figura 88

Configuración de aplicaciones.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'channels',
    'servicioNube',
    'servicioLocal',
]
```

En el archivo “asgi.py” se declaran las conexiones que se van a usar en la aplicación, por defecto se usa el protocolo http, y como se lo ha ido comentando en el desarrollo del proyecto websocket también es un protocolo, por lo que se lo debe añadir en el archivo, en la figura 89 se muestra como agregar este protocolo, cabe recalcar que el archivo “asgi.py” permite que existan métodos asincrónicos.

Figura 89

Configuración del archivo asgi.py.

```
import os
from django.core.asgi import get_asgi_application
from channels.auth import AuthMiddlewareStack
from channels.routing import ProtocolTypeRouter
from channels.routing import URLRouter
from Clasificador.routing import ws_urlpatterns

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Clasificador.settings')

application = ProtocolTypeRouter({
    'http': get_asgi_application(),
    'websocket': AuthMiddlewareStack(URLRouter(ws_urlpatterns))
})
```

Una vez configurado los protocolos, se crea el archivo “routing.py”, en el cual se agregan las direcciones URL para realizar las conexiones de WebSocket con el consumidor como se lo puede observar en la figura 90.

Figura 90

URLs de conexión WebSocket.

```
from django.urls import path
from servicioNube.views import WSConsumerAzure
from servicioLocal.views import WSConsumerResnet

ws_urlpatterns = [
    path('ws/redAzure/', WSConsumerAzure.as_asgi()),
    path('ws/redResnet/', WSConsumerResnet.as_asgi()),
]
```

Después de haber agregado los protocolos en el archivo, se debe configurar el canal Redis, para poder habilitar la comunicación, como se muestra en la figura 91.

Figura 91

Configuración de canal.

```
CHANNEL_LAYERS={
    'defauld':{
        'BACKEND':'channels_redis.core.RedisChannelLayer',
        'CONFIG':{
            'host':[('127.0.0.1')]
        }
    }
}
```

Desde el programa principal se envía los datos que se cree que son relevantes para el usuario, tanto la interfaz con el servicio en la nube como el del servicio local cuentan con una estructura de programación similar para el envío de datos a través del formato JSON, en la

figura 92 se muestra el envío de datos del servicio en la nube y en la figura 93 se muestra el envío de datos del servicio local, el envío de datos se realiza de manera asíncrona.

Figura 92

Envío de dato al servicio en la nube.

```
class WSConsumerAzure(AsyncWebsocketConsumer):
    async def connect(self):
        global decSensor
        decSensor=True
        await self.accept()
        prediccionRed1=""
        while True:
            if (GPIO.input(22) == 1): #Detectar si la banda esta prendida
                if decSensor==True:
                    if (GPIO.input(23) == 0):
                        decSensor=False
                        apagarBanda()
                        prediccionRed1=idenObjeto() #Retorna si es Obj1, Obj2 o
ninguno y enciende la banda
                    if(GPIO.input(25)==0):
                        apagarServo(predicAzure,Obj1Az,Obj2Az)
                        datosNube = {'obj1Nube':Obj1Az, 'obj2Nube':Obj2Az,
                                'cont1Nube':contObj1, 'cont2Nube':contObj2,
                                 'cont3Nube':contNing,
                                'confNube':valConfA, 'conf1Nube':valConfA1,
                               'conf2Nube':valConfA2,
                                'predNube':predicAzure,'estado':estado,'servoM1':
                                 servoM1,'servoM2':servoM2,
                                'banda':banda,'foto':urlImg}
                        r = requests.post(url,data=datosNube)
                elif (GPIO.input(23) == 1):
                    decSensor=True
            datosEnv={'message':prediccionRed1,'fotoA':dirFoto,'fotoAz2':dirFoto2,
             'banF':bandImg,'contAzObj1':contObj1,'contAzObj2':contObj2,'contNing'
      :contNing,'valApredic':valConfA,'valApredic1':valConfA1,'valApredic2':valCo
      nfA2,'obj1':Obj1Az, 'obj2':Obj2Az,'servoM1':servoM1,'servoM2':servoM2,
             'banda':banda}
            await self.send(json.dumps(datosEnv))
            await sleep(0.1)
```

Figura 93

Envío de datos al servicio local.

```
class WSConsumerResnet(AsyncWebsocketConsumer):
    async def connect(self):
        global decSensor,servoM1,servoM2,banda
        decSensor=True
        await self.accept()
        prediccionRed2=""
        while True:
            if (GPIO.input(22) == 1):
                if decSensor==True:
                    if (GPIO.input(23) == 0):
                        decSensor=False
                        banda = 0
                        apagarBanda()
                        prediccionRed2=PrediccionResN() #Retorna si es Obj1, Obj2
o ninguno y enciende la banda
                    if(GPIO.input(25)==0):
                        servoM1 = 0
                        servoM2 = 0
                        apagarServo(prediccionR,Obj1ReN,Obj2ReN)
                        datosLocal = {'obj1Local':Obj1ReN, 'obj2Local':Obj2ReN,
                            'cont1Local':rContObj1, 'cont2Local':rContObj2,
                            'confLocal':valPredic, 'conf1Local':valPredic1,
                           'conf2Local':valPredic,
                            'predLocal':prediccionR,'estado':estado,'foto':urlImg,
                           'servoM1':servoM1,'servoM2':servoM2,'banda':banda}
                        r = requests.post(urlL,data=datosLocal)
                elif (GPIO.input(23) == 1):
                    decSensor=True
            datosRl={'messageR':
prediccionRed2,'fotoR':dirFotoRes,'fotoR2':dirFotoRes2,'banFr':bandFr,
            'contResnObjR1':rContObj1,'contResnObjR2':rContObj2,
            'valRpredic':valPredic,
            'valRpredic1':valPredic1,'valRpredic2':valPredic2,'aviso': aviso,
            'obj1':Obj1ReN, 'obj2':Obj2ReN,
            'bandEntren':bandEntren,'bandCargar':bandCargar,
            'bandDatosModel':bandDatosModel,'bandDatos':bandDatos,
            'bandBorrar':bandBorrar,'bandCarpeta':bandCarpeta,
            'bandInicio':bandInicio,
            'servoM1':servoM1,'servoM2':servoM2,'banda':banda}
            await self.send(json.dumps(datosRl))
            await sleep(0.1)
```

Una vez enviado los datos se debe receptar en el servidor web, por lo que se crea un archivo de JavaScript, el cual permita conectar con el archivo de Python y realizar la comunicación de los datos, en la figura 94 se muestra la conexión del enlace con la interfaz del servicio en la nube y se almacena en la variable socket.

Figura 94

Enlace de conexión al servicio en la nube.

```
var socket = new WebSocket('ws://localhost:8000/ws/redAzure/')
```

Posteriormente para la recepción de datos se realiza la conexión mediante la subfunción “onmessage”, la cual capta la información que se ha enviado por el formato JSON, y se almacena en diferentes variables, permitiendo mostrar la información en la interfaz HTML diseñada previamente, como lo ilustra la figura 95.

Figura 95

Captación de datos del servicio en la nube.

```
socket.onmessage = function (event) {
  var data = JSON.parse(event.data);
  if (data.banF) {
    document.querySelector('#predicAzure').innerText = data.message;
    document.getElementById("imgAzure").src = data.fotoA;
    document.getElementById("imgAzure2").src = data.fotoAz2;
    document.querySelector('#contAzObj1').innerText = data.contAzObj1;
    document.querySelector('#contAzObj2').innerText = data.contAzObj2;
    document.querySelector('#contAzNing').innerText = data.contNing;
    updatechart(data.valApredic); //actualiza gráfico de pastel
    if (contGraf1 != data.contAzObj1) { //actualiza gráfico lineal 1
      actualizarGraf1(data.obj1,data.contAzObj1, data.valApredic1);
      contGraf1 = data.contAzObj1; }
    if (contGraf2 != data.contAzObj2) { //actualiza gráfico lineal 2
      actualizarGraf2(data.obj2,data.contAzObj2, data.valApredic2);
      contGraf2 = data.contAzObj2; }
  }
  actuadores(data.banda,'idEstadoBanda',"ENCENDIDA","APAGADA");//estado de banda
  actuadores(data.servoM1,'idEstadoServ1',"CERRADA","ABIERTA");//estado de servo 1
  actuadores(data.servoM2,'idEstadoServ2',"CERRADA","ABIERTA");//estado de servo 2
}
```

De igual forma para la conexión con el servicio local, se crea otro archivo JavaScript el cual contenga los datos únicamente del servicio local, debido a esto se ha creado diferentes enlaces, en la figura 96 se puede observar el enlace para conectarse mediante WebSocket

Figura 96

Enlace de conexión al servicio local.

```
var socket = new WebSocket('ws://localhost:8000/ws/redResnet/')
```

Una vez enlazado con el servidor se procede a captar los datos enviados desde el archivo Python, y se los almacena en diferentes variables, posteriormente se muestra esa información en la página web del servicio usado, como lo muestra la figura 97.

Figura 97

Captación de datos del servicio local.

```
socket.onmessage = function (event) {
  var data = JSON.parse(event.data);
  document.querySelector('#idAvisos').innerText = data.aviso;
  if(data.bandDatos){
  }else if(data.bandDatosModel){
  }else if(data.bandCargar){
  } else if(data.bandBorrar){
  }
  if(data.bandEntren){
  }
  if(data.bandCarpeta){
  }
  if(data.bandInicio){
  }
  document.querySelector('#predicResnet').innerText = data.messageR;
  document.querySelector('#contRlObj1').innerText = data.contResnObjR1;
  document.querySelector('#contRlObj2').innerText = data.contResnObjR2;
  if (data.banFr) {
    }
  actuadores(data.banda,'idEstadoBanda',"ENCENDIDA","APAGADA");//estado de banda
  actuadores(data.servoM1,'idEstadoServ1',"CERRADA","ABIERTA");//estado de servo 1
  actuadores(data.servoM2,'idEstadoServ2',"CERRADA","ABIERTA");//estado de servo 2
}
```

Interfaz Remota

Para la interfaz remota es necesario las mismas librerías que se usaron para la interfaz local, los pasos para poder usar este protocolo son similares al que se explicó previamente, la diferencia se encuentra en la forma de obtener los datos, para poder actualizar los datos desde la Raspberry, se debe hacer uso de la librería csrf_exempt, la cual permite ingresar información de forma remota, caso contrario el servido envía un código de estado de error, para usar esta librería se debe colocar al inicio de la función, en la figura 98 se puede visualizar su uso en el servicio en la nube, y en la figura 99 el uso en el servicio local.

Figura 98

Captación de datos de la interfaz remota del servicio en la nube.

```
@csrf_exempt
def Nube(request):
    global obj1Nube, obj2Nube, cont1Nube, cont2Nube, cont3Nube, confNube,
conf1Nube, conf2Nube
    global predNube, estado, foto, servoM1, servoM2, banda
    if request.method == "POST":
        obj2Nube = request.POST['obj2Nube']
        obj1Nube = request.POST['obj1Nube']
        cont1Nube = request.POST['cont1Nube']
        cont2Nube = request.POST['cont2Nube']
        cont3Nube = request.POST['cont3Nube']
        confNube = request.POST['confNube']
        conf1Nube = request.POST['conf1Nube']
        conf2Nube = request.POST['conf2Nube']
        predNube = request.POST['predNube']
        estado = request.POST['estado']
        servoM1 = request.POST['servoM1']
        servoM2 = request.POST['servoM2']
        banda = request.POST['banda']
        foto = request.POST['foto']
    return render(request, "datosNube.html")
```

Figura 99

Captación de datos de la interfaz remota del servicio en la local.

```
@csrf_exempt
def Local(request):
    global obj1Local, obj2Local, cont1Local, cont2Local, confLocal,
conf1Local, conf2Local, predLocal
    global estado, foto, servoM1, servoM2, banda
    if request.method == "POST":
        obj2Local = request.POST['obj2Local']
        obj1Local = request.POST['obj1Local']
        cont1Local = request.POST['cont1Local']
        cont2Local = request.POST['cont2Local']
        confLocal = request.POST['confLocal']
        conf1Local = request.POST['conf1Local']
        conf2Local = request.POST['conf2Local']
        predLocal = request.POST['predLocal']
        estado = request.POST['estado']
        servoM1 = request.POST['servoM1']
        servoM2 = request.POST['servoM2']
        banda = request.POST['banda']
        foto = request.POST['foto']
    return render(request, "datosLocal.html")
```

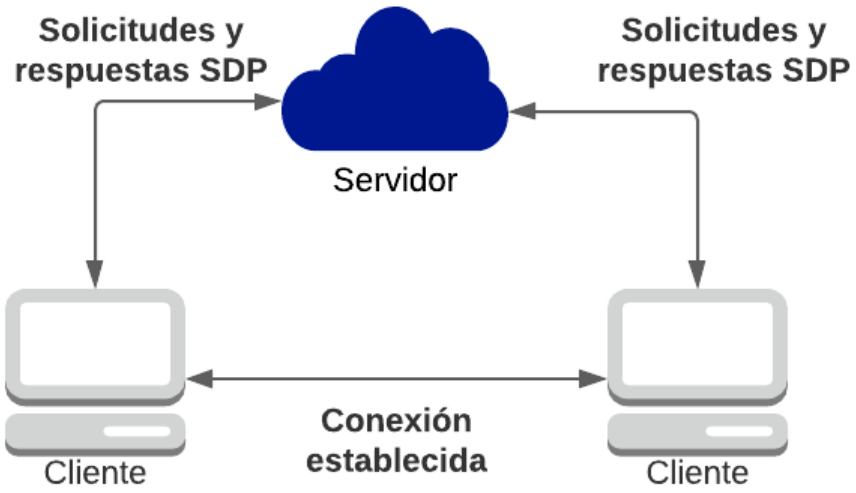
Web RTC

Al crear un canal de conexión es importante poder transmitir en tiempo real la información, por lo que se ha usado la tecnología Web RTC, que permite la comunicación en tiempo real, gracias a esta característica se puede actualizar la información y mostrar al usuario los datos más recientes obtenidos. En la figura 100 se puede observar cómo se transmiten los datos, antes de conectarse entre clientes, primero se envía una solicitud SDP (Protocolo de Descripción de Sesión) al servidor, en esta solicitud se encuentra la dirección del cliente para la comunicación, posteriormente el servidor envía esa información al otro cliente para que pueda conocer la dirección a quien debe transmitir los datos, de la misma forma el segundo cliente también envía mediante un servidor sus datos para que el primer cliente recepte esa

información y establezcan una conexión, una vez que los dos clientes conocen el destino a la cual realizar el envío de datos, ya no es necesario una conexión al servidor, debido a que se lo hace directamente entre los clientes, de esta manera permite la simplificación de la comunicación al evitar un servidor.

Figura 100

Estructura de WebRTC.



Interfaz Local

Para la comunicación de video se lo programa en un archivo JavaScript, el cual permite mostrar la imagen de la cámara en tiempo real, esto facilita al usuario observar el funcionamiento del sistema clasificador, primero se debe identificar el ID del dispositivo a usar, una vez que haya sido detectado se procede a realizar la comunicación con el mismo, de existir un error se notifica en la consola, este código se lo puede observar en la figura 101.

Figura 101

Función de streaming.

```
function startStreaming() {
  idcamera = 0
  for (var i = 0; i < myVideoInputs.length; i++) {
    if (myVideoInputs[i].label == "Integrated Camera (0c45:64ab)") {
      idcamera = i;
      break;
    }
  }
  var mediaSupport = 'mediaDevices' in navigator;
  if (mediaSupport && null == cameraStream) {
    navigator.mediaDevices.getUserMedia({
        video: {
          deviceId: myVideoInputs[idcamera].deviceId
        }
      })
      .then(function (mediaStream) {
        cameraStream = mediaStream;
        stream.srcObject = mediaStream;
        stream.play();
      })
      .catch(function (err) {
        console.log("Acceso denegado a la cámara: " + err);
      });
  } else {
    alert('Está en uso la cámara.');
    return;
  }
}
```

Activación de actuadores. La activación de los servomotores es similar tanto para el servicio local como el de la nube, en la figura 102 se puede observar la forma de cómo se activan los actuadores, realiza una comparación entre la predicción realizada con los datos del objeto uno o dos, para poder activar uno de los servomotores.

Figura 102

Función de activación de actuadores.

```
for i in etiqueta:
        if (i == Obj1):
            servoM1 = 1
            servo1.ChangeDutyCycle(2)#Activacion servo1 90°
            time.sleep(0.5)
            servo1.ChangeDutyCycle(0)
            predicAzure = Obj1
            valConfA=confiabilidad[conTag]
            valConfA1=valConfA
            global contObj1
            contObj1=contObj1+1
            contador = contObj1
            break
        elif(i == Obj2):
            servoM2 = 1
            servo2.ChangeDutyCycle(2)#Activacion servo2 90°
            time.sleep(0.5)
            servo2.ChangeDutyCycle(0)
            predicAzure = Obj2
            valConfA=confiabilidad[conTag]
            valConfA2=valConfA
            global contObj2
            contObj2=contObj2+1
            contador = contObj2
            break
```

Conexión con base de datos. Para el enlace con la base de datos Firebase se debe instalar las librerías con el comando

pip install pyrebase4

Una vez instalada la librería para hacer el uso de la base de datos en la nube, a continuación, se abre el script en el cual se desea usar este servicio, por lo que primero hay que realizar la importación de la librería, el comando para esto es:

import pyrebase

Posteriormente se configuran las credenciales del servicio, esto permite verificar que exista el servicio al cual se necesita enlazar, estos datos se los adquiere desde el proyecto creado en Firebase como muestra la figura 103.

Figura 103

Configuración de base de datos.

```
#---Configuracion base de datos
config = {
    "apiKey": "Valor de apiKey",
    "authDomain": "Dirección del dominio",
    "databaseURL": "Dirección de la base de datos",
    "projectId": "ID del proyecto",
    "storageBucket": "Dirección del storage",
    "messagingSenderId": "ID de mensaje",
    "appId": "ID de la aplicación",
}
```

Una vez ingresadas las credenciales se procede a autentificar los datos, mediante las variables como storage o database como se muestra en la figura 104, se puede acceder a los servicios de Firebase en la nube.

Figura 104

Asignación de información.

```
firebase = pyrebase.initialize_app(config)
authe = firebase.auth()
database = firebase.database()
storage = firebase.storage()
```

Para el envío de datos de una manera ordenada y correcta, primero se crea un diccionario con los valores necesarios que son necesarios guardar, a continuación, mediante la variable en la que se almacenó el acceso a la base de datos se procede a ingresar los valores previamente guardados, la organización de para guardar los datos comienza desde una tabla en general llamada Datos, y se divide en tablas por fechas, cada una de esta subtabla cuenta

con divisiones de procesos, y cada proceso tiene una segmentación por horas de identificación en las cuales dentro de estas se puede notar los valores leídos en dicho momento, como lo ilustra la figura 105.

Figura 105

Envío de información a la base de datos.

```
data={"nombre":nombre,'contador':contador,'porcentaje':porcentaje}
    database.child('Datos').child(fecha).child(procesoA).child(hora).set(data)
```

Alojamiento de la aplicación en Heroku. Para realizar el proceso de subir la aplicación a la plataforma, primero es necesario instalar el CLI (interfaz de línea de comandos) de Heroku, esto permite que desde un terminal se pueda crear y administrar las aplicaciones que se van a subir a la plataforma.

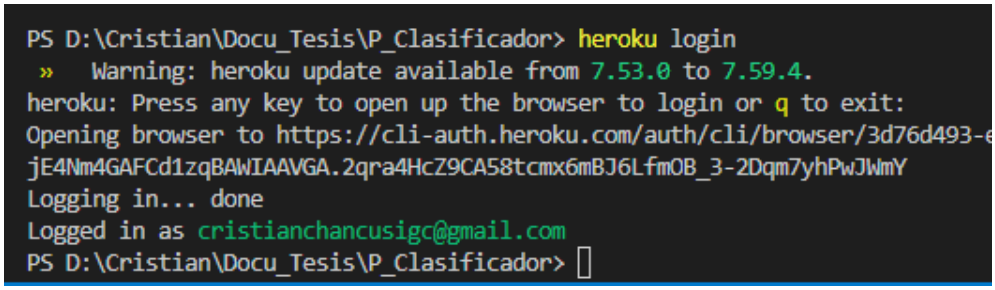
Una vez instalada esta herramienta, se procede a ejecutar el siguiente comando en el terminal:

heroku login

A continuación, se abrirá automáticamente un navegador en la página de registro de Heroku, inicia sesión y si el proceso es correcto se mostrará un resultado como en el que se puede observar en la figura 106.

Figura 106

Mensaje de éxito.



Luego de haber creado una aplicación en la plataforma de heroku, se procede a crear un repositorio Git, este se debe encontrar al nivel del archivo “manage.py”, para iniciar el repositorio se debe ingresar el siguiente comando en la consola:

git init

heroku git:remote -a sistemaclasificador

Para poder subir el proyecto a la nube, se requiere la instalación de librerías como: psycopg2, Daphne, decouple y whitenoise, esto se lo hace mediante el comando “pip install” en la terminal.

La aplicación en la nube necesita conocer de las librerías que se usan, por lo que es importante obtener un archivo en el cual conste los requerimientos usados, por lo que en la terminal se ingresa el siguiente comando:

pip freeze > requirements.txt

A continuación, se procede a la configuración del archivo “settings.py”, primero se modifica el parámetro “ALLOWED_HOSTS”, el cual indica los dominios permitidos para el ingreso de la aplicación, y también se debe configurar la variable de los orígenes confiables con la dirección de la aplicación web, las configuraciones se muestran en la figura 107.

Figura 107

Configuración de orígenes confiables.

```
ALLOWED_HOSTS = ["*"]
CSRF_TRUSTED_ORIGINS = ['https://*.sistemaclasificador.herokuapp.com/']
```

Para poder subir el archivo a la nube, es necesario la inclusión de la librería “whitenoise” como se lo muestra en la figura 108.

Figura 108

Configuración de librería.

```
MIDDLEWARE = [
    'whitenoise.middleware.WhiteNoiseMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Es necesario la creación de los archivos estáticos como lo muestra la figura 109.

Figura 109

Configuración de archivos estáticos.

```
STATICFILES_STORAGE = "whitenoise.storage.CompressedManifestStaticFilesStorage"
```

Posteriormente se crea el archivo “Procfile” a nivel del “manage.py”, este archivo es necesario debido a que, al momento de iniciar el programa en la nube, este será el primero en ejecutarse, la configuración del “Procfile” es el que se ilustra en la figura 110.

Figura 110

Archivo Procfile.

Procfile

```
1   web: daphne P_Clasificador.asgi:application --port $PORT --bind 0.0.0.0 -v2
2   worker: python manage.py runworker --settings=P_Clasificador.settings -v2
```

Una vez realizado todas las respectivas configuraciones, se procede a subir el código al repositorio previamente creado, primero se guarda todos los cambios por lo que mediante el terminal se ingresa el siguiente comando:

git add .

A continuación, se procede a subir los cambios del código al repositorio mediante el comando:

git comit -am “subida”

Finalmente se sube los cambios a la plataforma de heroku, esto se lo realiza ejecutando el comando:

git push heroku master

Una vez finalizada la ejecución del comando, si todo se encuentra en orden, en la terminal se mostrará un mensaje de subida exitosa y la URL de la aplicación, la dirección URL para la supervisión del proceso de este proyecto es:

https://sistemaclasificador.herokuapp.com/

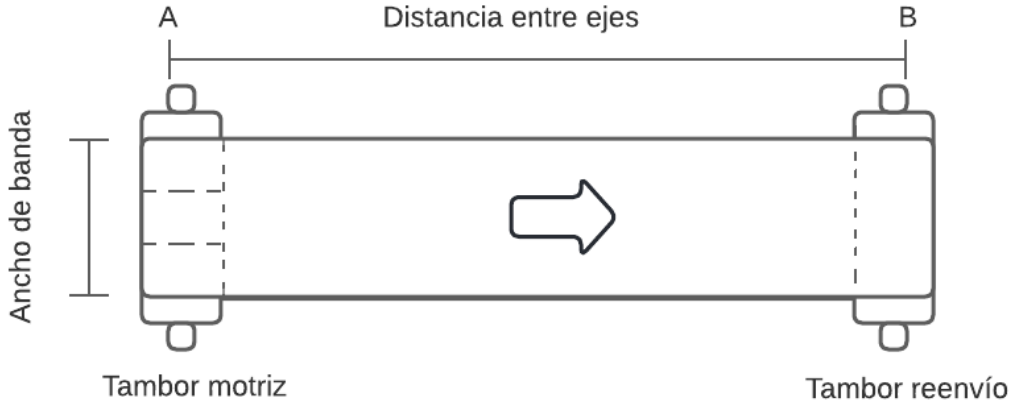
Capa Física

Estructura mecánica

Como se mencionó en el capítulo de diseño, se emplea una estructura del tipo banda transportadora, que permita el ingreso de manera serial de los objetos. Bajo la guía y consideraciones de diseño y construcción que se mencionan en Forbo, (2022) se implementa una banda estándar para transporte ligero, esta banda marcha sobre dos tambores situados en los extremos, un tambor motriz y un tambor de reenvío, como se ilustra en la figura 111 se considera para la distancia entre ejes un valor de 64 cm, para el ancho de la banda un valor de 10.5 cm, el tambor matriz estará conectado al motor y estará ubicado a la izquierda y entrada del sistema.

Figura 111

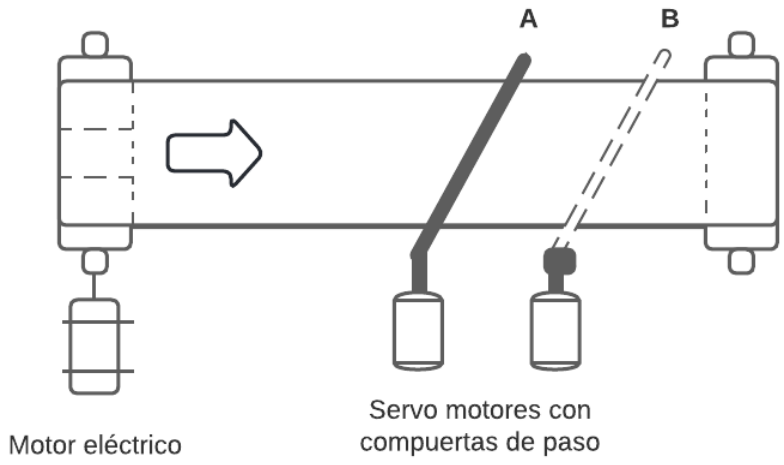
Estructura mecánica.



Otro aspecto importante en la implementación de la estructura mecánica, es la ubicación de los actuadores electromecánicos, por lo que se distribuyen de acuerdo a la figura 112, el motor eléctrico se conectará al tambor motriz, permitiendo así el movimiento de la banda, por otro lado, a cada uno de los dos servomotores se les incorporará una compuerta diagonal que permitirá la clasificación de los objetos ingresados en dos grupos, en el estado presentado en la figura, la compuerta A está cerrada y con la banda en movimiento hará que el objeto se desvíe y caiga en su grupo, mientras que la compuerta B en estado de apertura permitiría el paso de un objeto. En caso de que el objeto clasificado pertenezca al grupo A el estado de la compuerta A será cerrado y de la compuerta B será abierta, si el objeto pertenece al grupo B sucederá lo contrario la compuerta A permitirá el paso en estado de apertura y la compuerta B se cerrará, finalmente si el objeto no pertenece a ninguno de los dos grupos, ambas compuertas permanecerán abiertas.

Figura 112

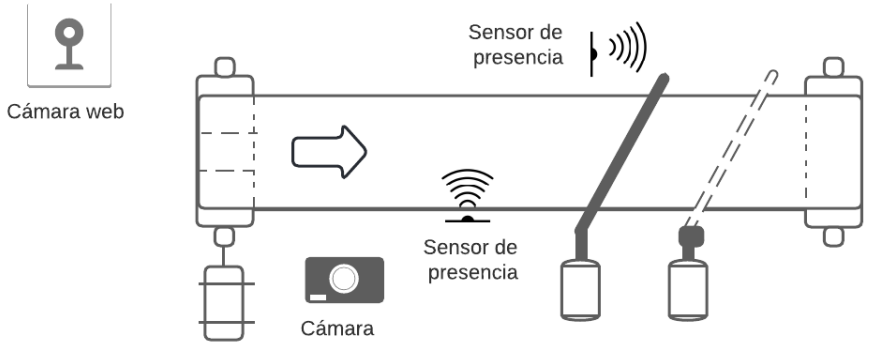
Ubicación de actuadores.



Finalmente, para la distribución de la estructura es importante determinar la posición de los sensores y cámaras. Los sensores de presencia tendrán dos funciones, la primera será detectar el objeto en medio recorrido para así dar una señal de paro a la banda y una señal de captura a la cámara para que inicie el proceso de análisis y predicción del objeto, por lo que, se lo ubica a la izquierda de los servos motores a medio recorrido de la banda, por su parte el segundo sensor de presencia será el encargado de detectar la salida del objeto una vez ha sido clasificado y separado en grupos por las compuertas, este dará la señal de apertura de la compuerta que se ha cerrado según el caso. La cámara permitirá la captura de la imagen del objeto para su análisis, por eso está ubicada a la entrada de la banda, por otra parte, la cámara web será la encargada de transmitir en tiempo real el funcionamiento del proceso y estará ubicada en una esquina superior que permita la visualización del mismo. Así la distribución de los sensores y actuadores quedan como lo muestra la figura 113.

Figura 113

Distribución de sensores y actuadores.



Circuito electrónico

El circuito está compuesto de un microordenador para el procesamiento de datos y control general del circuito, por sensores de presencia, dos cámaras, dos servomotores y un motor DC, además de las fuentes de alimentación necesarias para cada elemento. Por lo que a continuación se detallan cada uno de ellos y su modo de funcionamiento y conexión dentro del proyecto.

RaspBerry Pi 4B. El Microordenador en su versión de 4GB de memoria RAM presenta las características y especificaciones técnicas detalladas en la tabla 13 y tomadas de RaspberryPi, (2022).

Tabla 13

Especificaciones de Raspberry Pi 4B.

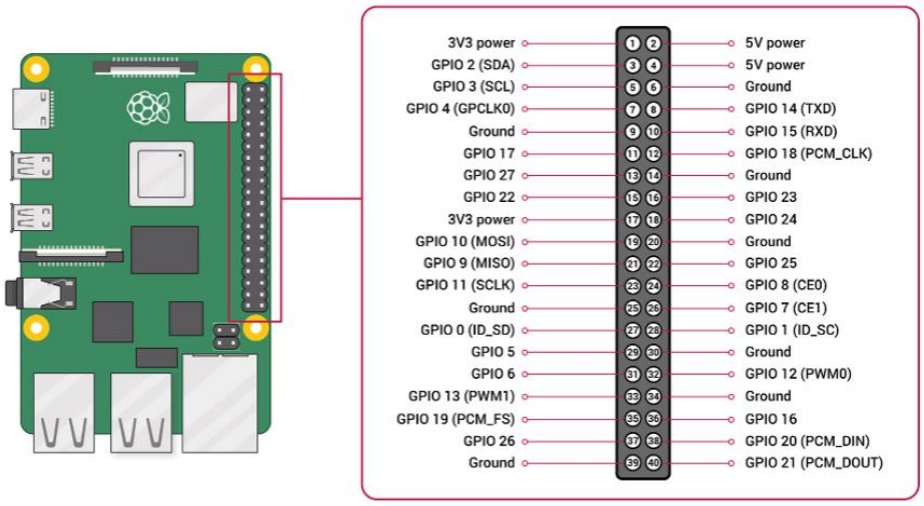
Especificaciones	Descripción
Procesador	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
Memoria	1GB, 2GB, 4GB o 8GB LPDDR4 con on-die ECC

Especificaciones	Descripción
Conectividad	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports
GPIO	Puerto estándar de 40 pines
Video y sonido	2 puertos micro HDMI (soporta 4Kp60) 2-lane MIPI DSI display 2-lane MIPI CSI camera puerto 4-polos estéreo audio.
Multimedia	H.265 (4Kp60 decodificador); H.264 (1080p60 decodificar, 1080p30 codificador); OpenGL ES, 3.0 gráficos
Tarjetas SD	Micro SD.
Alimentación	5V DC vía USB-C conector (mínimo 3A) 5V DC vía GPIO (mínimo 3A) Alimentación a través de Ethernet (PoE) habilitada (requiere SOMBRERO PoE separado)
Ambiente	Temperatura de operación 0-50°C

Para el desarrollo del proyecto es necesario conocer la distribución de pines y del GPIO, a través de él se procede al uso de sus recursos de pines de lectura y escritura, así como de fuente de alimentación con tensiones de 3.3 y 5 V DC. La figura 114 ilustra la distribución empleada para la conexión del circuito electrónico.

Figura 114

Pines de Raspberry Pi 4.

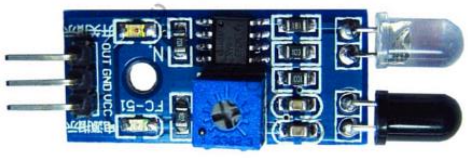


Nota: Tomada de GPIO por Hardwarelibre, https://www.hwlibre.com/gpio-raspberry-pi/

Sensor FC51. El módulo de sensor de obstáculos permite la detección de objetos según su proximidad por medio de la emisión de energía infrarroja IR y la captación por medio de un receptor la figura 115 ilustra el módulo empleado.

Figura 115

Sensor FC51.



Nota: Tomado de carrod electrónica, https://www.carrod.mx/products/modulo-sensor-infrarojo-detector-de-obstaculos

La tabla 14 muestra las especificaciones técnicas del módulo FC51.

Tabla 14

Especificaciones de Sensor FC51.

Especificaciones	Descripción
Modelo	FC-51
Chip	LM393
Alimentación	3.3V – 5V
Voltaje de salida	5V
Distancia de detección	20 mm – 300 mm
Ángulo de detección	35°

Servomotor SG90. Para el proyecto se requiere acoplar una barrera metálica que permita la labor de compuertas, por lo que con el uso de dos servomotores SG90 se puede cumplir tal propósito, la figura 116 muestra el dispositivo empleado.

Figura 116

Servomotor.

Nota: Tomado de Unit electronic, https://uelectronics.com/producto/servomotor-sg90-rc-9g/

La tabla 15 detalla las especificaciones técnicas de los dos servomotores.

Tabla 15

Especificaciones de servomotor.

Especificaciones	Descripción
Alimentación	+5V DC
Par de torsión	2.5kg/cm
Velocidad	0.1s/60°
Rotación	0° - 180°
Engranaje	Plástico
Peso del motor	9g

Motor reductor DC. El motor permite el control de velocidad para proyectos en los que la velocidad y la fuerza son un factor clave, para este caso es necesario para la conexión con el tambor motriz y movimiento de la banda transportadora. La figura 117 ilustra el motor empleado.

Figura 117

Motor reductor.

Nota: Tomado de DeasTronic, https://www.ideastronic.com/producto/motor-reductor-dc-12v-60rpm-jga-25-370/

La tabla 16 detalla las especificaciones técnicas del motor reductos DC12V 60RPM.

Tabla 16

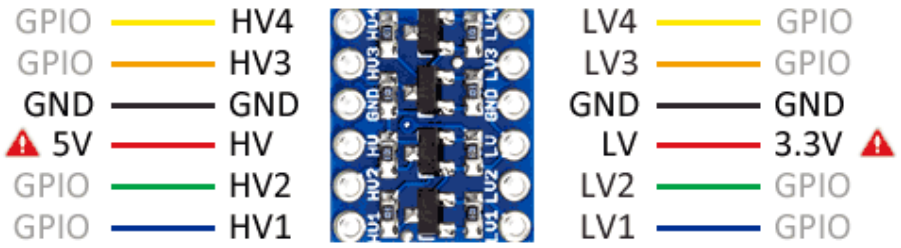
Especificaciones del motor.

Especificaciones	**Descripción**
Tensión de funcionamiento	6V – 18V Pico
Tensión nominal	12V
Velocidad sin carga	60 RPM
Corriente a 12V	Sin carga 50mA
	Media carga
	240mA
	Pico 1.2 A
Torque	4.78kg-cm
Relación engranaje	171:1
Engranaje	Plástico
Peso del motor	91g

Adaptador de nivel. Debido a la incompatibilidad de nivel lógico manejado por los sensores y el microordenador es necesario emplear un Level Shifter bidireccional que permita la conversión de TTL 5V entregado por los sensores y módulos a 3.3V de nivel lógico I2C manejado por la Raspberry Pi. Se emplea un módulo conversor como el mostrado en la figura 118, para el ingreso de un bajo voltaje de 3.3v y un alto voltaje de 5V, 4 pines con voltaje alto de HV1-HV4 y 4 pines de bajo voltaje LV1 – LV4.

Figura 118

Adaptador de nivel.

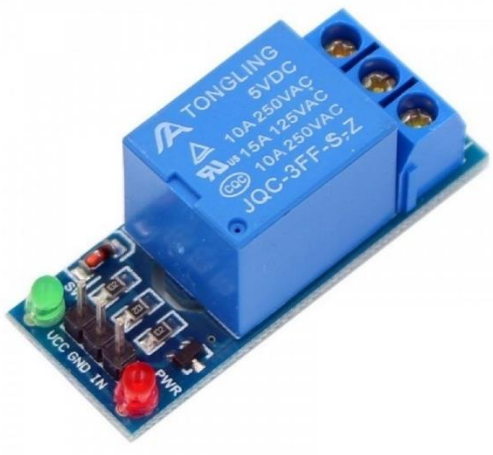


Nota: Tomado de solectro, https://solectroshop.com/es/convertidores-logicos/1282-modulo-convertidor-ttl-5v-33v-de-nivel-logico-i2c-bidireccional.html

Módulo relé 1 canal. Es necesario el uso de una fuente de 12V extra para alimentación del motor reductor y para control del mismo, se empela un módulo relé de un canal que permita el encendido y apagado del motor a través de las señales de control enviadas desde el microordenador, a continuación, en la figura 119 se ilustra el módulo empleado, y en la tabla 17 sus especificaciones.

Figura 119

Módulo de relé.



Nota: Tomado de Megatronic, https://megatronica.cc/producto/modulo-rele-5v-1-canal-con-led-indicadores/

Tabla 17

Especificaciones del relé.

Especificaciones	Descripción
Tensión de funcionamiento	5V DC
Señal de control	TTL (3.3V o 5V)
Modelo Relay	SRD-05VDC-SL-C
Capacidad máxima	10ª/250VAC
Corriente máxima	10 A (NO), 5 A (NC)
Tiempo de acción	10ms /5 ms

Cámara Rev 1.3. Para la captura de imágenes se emplea un módulo de cámara propio para raspberry pi, la figura 120 ilustra la cámara empleada.

Figura 120

Cámara Rev.

Nota: Tomado de Pi Supply: https://uk.pi-supply.com/products/raspberry-pi-camera-board-v1-3-5mp-1080p?lang=es

Es módulo posee las especificaciones técnicas detalladas en la tabla 18.

Tabla 18

Especificaciones de la Cámara Rev.

Especificaciones	Descripción
Imágenes	imágenes estáticas de 2592 x 1944 píxeles
Video	1080p @ 30fps, 720p @ 60fps y 640x480p 60/90
Módulo	Omnivision 5647 de 5MP
Conexión	Interfaz serial de cámara MIPI de 15 pines
Tamaño	20x25x9mm
Peso	3g

Cámara Web. Para la transmisión mediante stream se emplea el uso de una cámara web como la mostrada en la figura 121, la cual posee especificaciones de resolución 1280x720, formato de video MJPEG, YUY2, sensor de imagen CMOS ½.7, compatible con USB 2.0 y 3.0.

Figura 121

Cámara web.

Nota: Tomado de Novicompu, https://www.novicompu.com/camara-web-havit-hv-n5086/p?idsku=3899&cmp_id=13178496464&adg_id=124248877404&kwd=&device=c&gclid=C

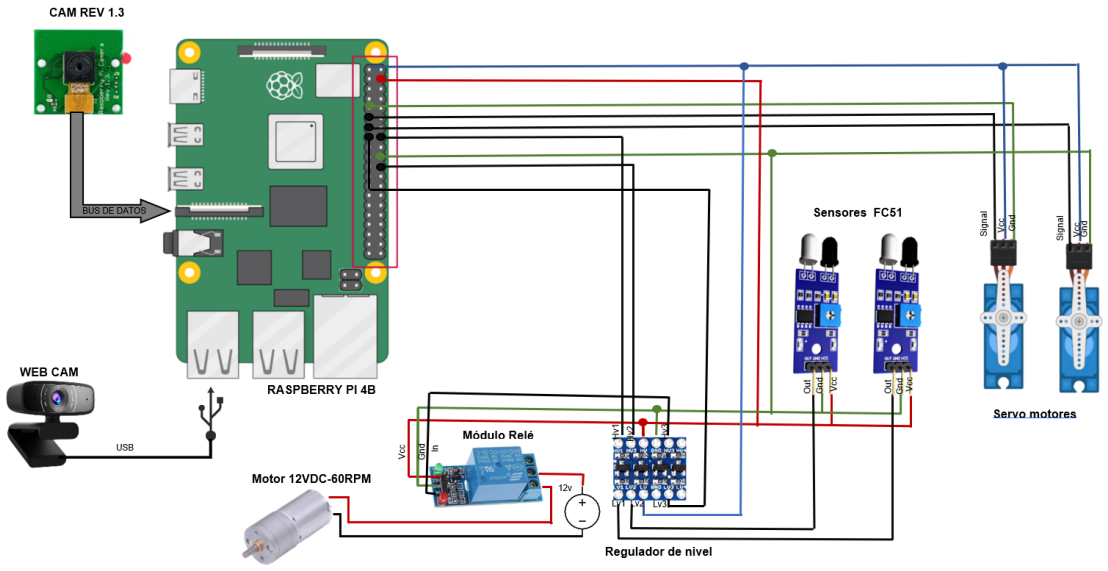
j0KCQjwwJuVBhCAARIsAOPwGARxRv98bkiuyOxH5hWUV23UDNDFMa5yg1jm0JntW4vrUQb VOQXCsvkaAunGEALw_wcB

Diagrama de conexión

Se ilustra en la figura 122 el diagrama de conexión de los elementos que conforman el sistema, los sensores de presencia FC51, los servomotores SG90, el motor reductor 12VDC, los módulos de conversión de tensión y de relé, así como la conexión de la cámara rev 1.3 y cámara web empeladas. Todos estos elementos conectados hacia el microordenador principal Raspberry Pi 4B.

Figura 122

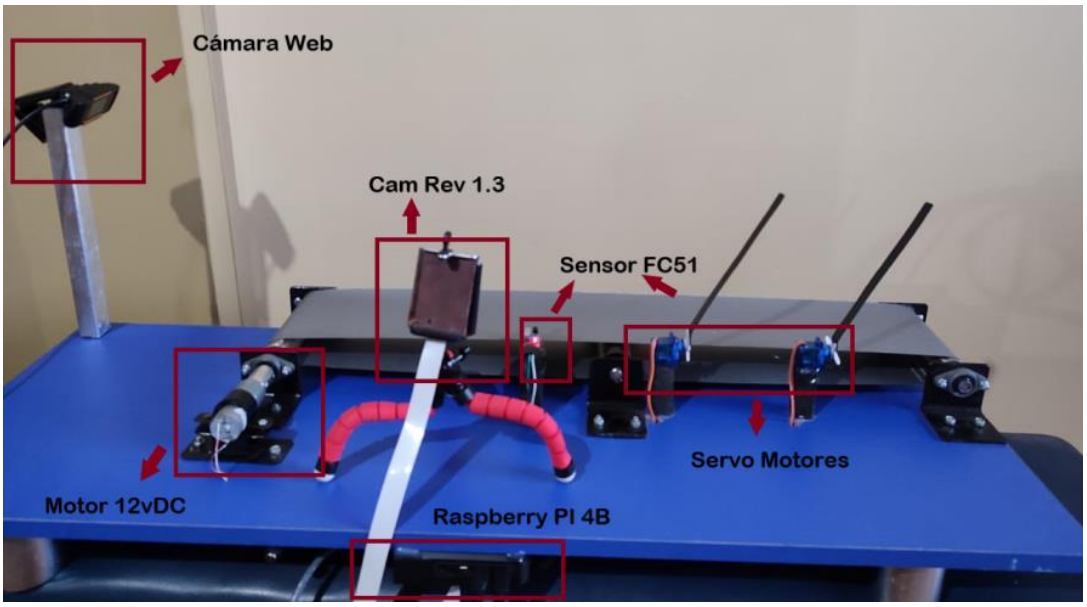
Diagrama de conexión.



Finalmente se muestra en la figura 123 la distribución implementada en una maqueta, la cual se basa en la guía mostrada por las figuras detalladas en la sección de la estructura mecánica.

Figura 123

Distribución de elementos.



Capítulo 5: Pruebas de Validación

Pruebas de Funcionamiento

Para validar el funcionamiento del sistema se realizan pruebas de clasificación de diferentes objetos empleando el servicio de la nube y el servicio local, para ello se determinan características de los objetos que son de interés al momento de emplear un sistema de clasificación, siendo estos la forma, el color y la textura de los objetos.

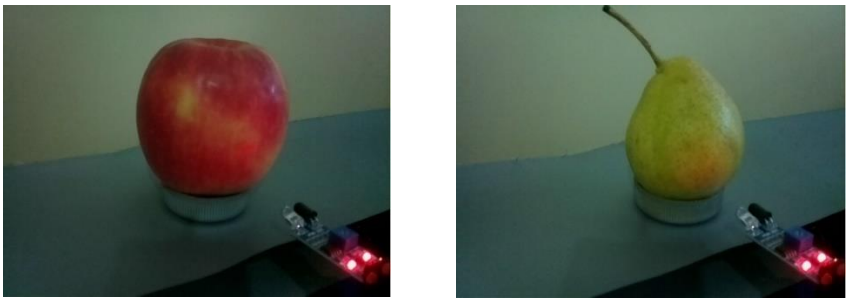
Pruebas del Servicio en la Nube

Pruebas de Forma. Para esto se establecieron 3 grupos de prueba, manteniendo la característica de que los objetos posean formas distintas en los dos primeros grupos y el tercero esté conformado por objetos de forma similar.

Prueba 1. Se emplean manzanas y peras como los mostrados en la figura 124.

Figura 124

Objetos de prueba 1.



Se realizan un total de 20 pruebas, 10 manzanas y 10 peras, obteniendo el resultado presentado en la tabla 19.

Tabla 19

Datos de prueba 1.

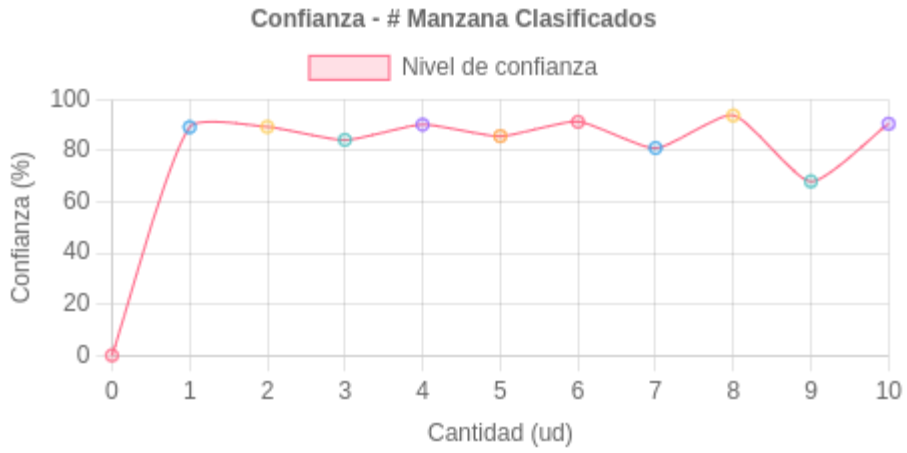
Num. de objeto	**Objeto ingresado**		**Predicción**		**Resultado de la predicción**	
	Manzana	***Pera***	***Manzana***	***Pera***	***Acierto [%]***	***Error [%]***
1	X		X		89.29	10.71
2	X		X		89.4	10.60
3	X		X		84.31	15.69
4	X		X		90.24	9.76
5	X		X		85.79	14.21
6	X		X		91.38	8.62
7	X		X		81.13	18.87
8	X		X		93.84	6.16
9	X		X		68.06	31.94

Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Manzana	***Pera***	***Manzana***	***Pera***	***Acierto [%]***	***Error [%]***
10	X		X		90.61	9.39
11		X		X	95.48	4.52
12		X		X	98.48	1.52
13		X	X		72.35	27.65
14		X		X	98.32	1.68
15		X		X	97.62	2.38
16		X		X	97.28	2.72
17		X		X	98.1	1.9
18		X		X	98.48	1.52
19		X		X	98.7	1.3
20		X		X	98.83	1.17

De la interfaz de control y supervisión se obtienen la figura 125, la cual muestra el nivel de confianza con el que el sistema predijo en cada iteración de manzanas, este está basado en el porcentaje de acierto de cada predicción, en él se aprecia que para los objetos “Manzana” se mantiene un acierto máximo de 91.38 % y mínimo de 68.06 %, sin embargo, el sistema clasificó de manera correcta las 10 manzanas.

Figura 125

Gráfica de confianza del objeto 1 de la prueba 1.



Para el caso de las peras se obtuvieron resultados presentados en la figura 126 en las que el porcentaje de confianza creció y se mantuvo en valores de entre 90.61 % y 98.83 % de acierto, sin embargo, para el objeto 13 presentado en la tabla anterior existió un falso negativo con un porcentaje de acierto del 72.35 % a pesar de ser una pera lo clasificó como manzana.

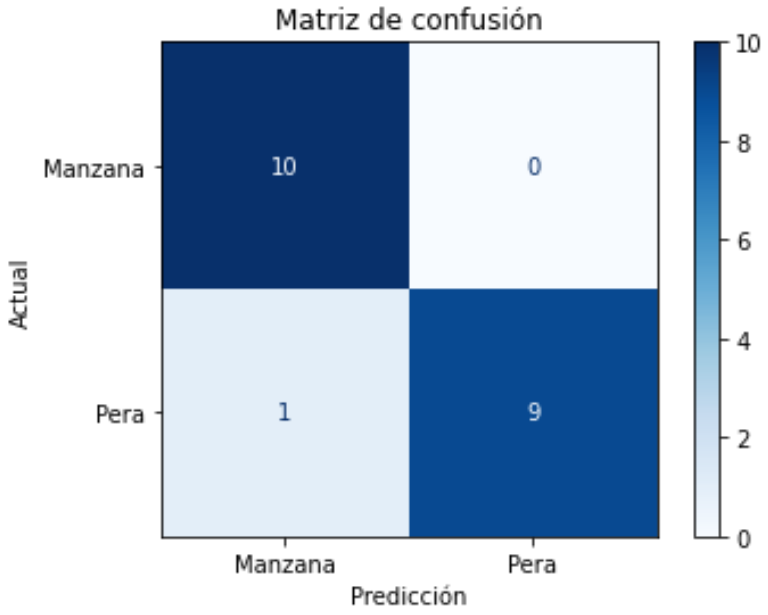
Figura 126

Gráfica de confianza del objeto 2 de la prueba 2.

Para conseguir una mejor interpretación y evaluación del modelo de predicción para esta prueba de clasificación se genera una matriz de confusión representada en la figura 127, obteniendo lo siguiente.

Figura 127

Matriz de la prueba 1.



En la matriz presentada se muestra un total de 20 imágenes de prueba con dos clases balanceadas, 10 imágenes para manzanas y 10 imágenes para peras, para la clase de manzanas existieron 10 verdaderos positivos, mientras que para la clase de peras existieron 9 verdaderos negativos y 1 falso positivo.

- Exactitud

$$Exactitud = \frac{VP + VN}{Total} = \frac{10 + 9}{20} = 0.95 \Longrightarrow 95\%$$

- Tasa de error

$$Tasa\ de\ error = \frac{FP + FN}{Total} = \frac{1}{20} = 0.05 \Longrightarrow 5\%$$

- Sensibilidad, exhaustividad

$$Sensibilidad = \frac{VP}{Total\ Positivos} = \frac{10}{0+10} = 1 \implies 100\%$$

- Especificidad

$$Especificidad = \frac{VN}{Total\ Negativos} = \frac{9}{1+9} = 0.9 \implies 90\%$$

- Precisión

$$Precisión = \frac{VP}{Total\ calsificados\ Positivos} = \frac{10}{1+10} = 0.91 \implies 91\%$$

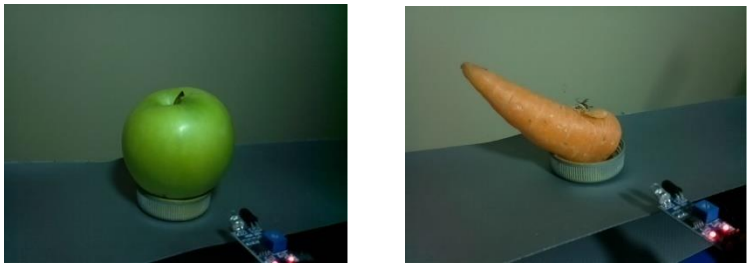
- Valor de predicción negativo

$$VPN = \frac{VN}{Total\ calsificados\ Negativos} = \frac{9}{9} = 1 \implies 100\%$$

Prueba 2. Para la segunda prueba se emplean formas distintas con frutas y verduras como los mostrados en la figura 128.

Figura 128

Objetos de prueba 2.



Se realiza un total de 20 pruebas, 10 frutas y 10 verduras, obteniendo el resultado presentado en la tabla 20.

Tabla 20

Resultados de la prueba 2.

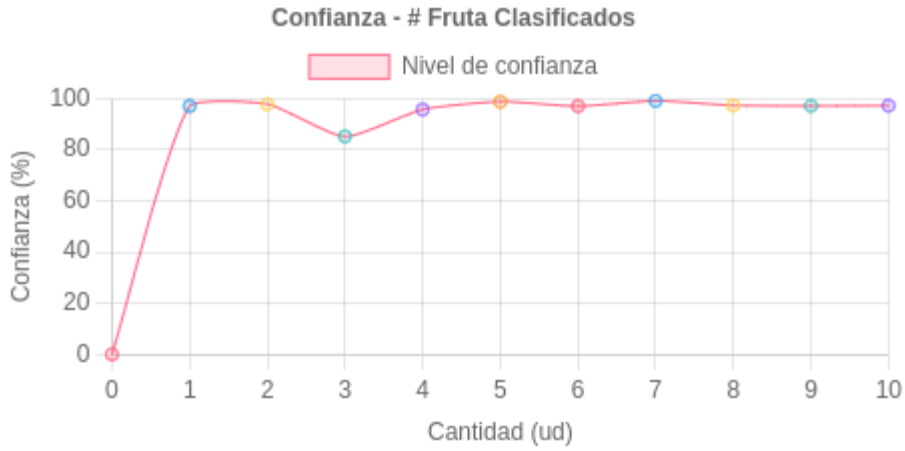
Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Frutas	***Vegetales***	***Frutas***	***Vegetales***	***Acierto [%]***	***Error [%]***
1	X		X		97.86	2.14
2	X		X		98.30	1.70
3	X		X		85.30	14.70
4	X		X		95.79	4.21
5	X		X		98.84	1.16
6	X		X		97.14	2.83
7	X		X		99.15	0.85
8	X		X		97.43	2.57
9	X		X		97.23	2.77
10	X		X		97.31	2.69
11		X		X	99.89	0.11
12		X		X	94.94	5.06
13		X		X	99.52	0.48
14		X		X	87.34	12.66
15		X		X	99.76	0.24
16		X		X	91.69	8.31
17		X		X	92.35	7.65
18		X		X	99.29	0.71
19		X	X		93.02	6.98

Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Frutas	***Vegetales***	***Frutas***	***Vegetales***	***Acierto [%]***	***Error [%]***
20		X		X	91.01	8.99

Se obtiene un resultado mostrado en la figura 129, esta muestra que las 10 frutas fueron clasificadas de manera correcta y con un porcentaje de confianza de entre 85.30 % al 99.15 %.

Figura 129

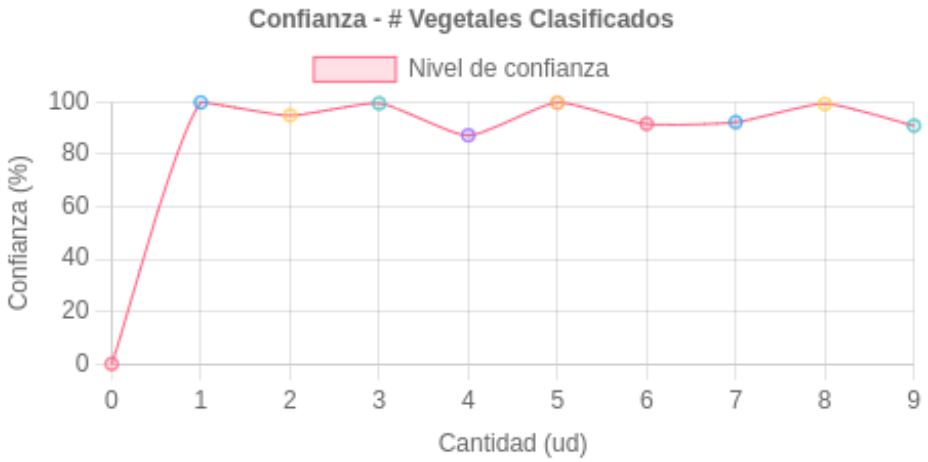
Gráfica de confianza del objeto 1 de la prueba 2.



Para el caso de los vegetales en el elemento 19 existió un falso positivo, pues se introdujo una cebolla y esta fue clasificada como manzana, sin embargo, los vegetales restantes fueron clasificados de manera correcta, como muestra la figura 132 con un nivel de confianza de entre 87.34 % al 99.29 %.

Figura 130

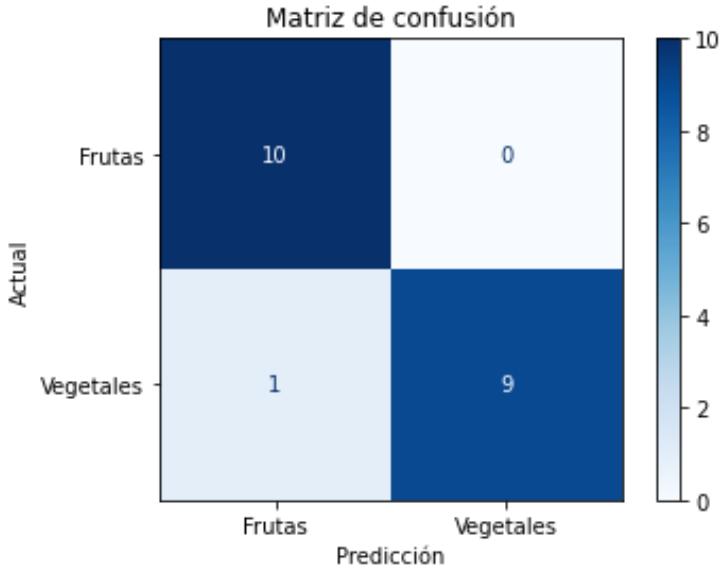
Gráfica de confianza del objeto 2 de la prueba 2.



Para conseguir una mejor interpretación y evaluación del modelo de predicción para esta prueba de clasificación se genera una matriz de confusión representada en la figura 131.

Figura 131

Matriz de la segunda prueba.



En la matriz presentada se muestra un total de 20 imágenes de prueba con dos clases balanceadas, 10 imágenes para frutas y 10 vegetales, para la clase de las frutas existieron 10

verdaderos positivos, mientras que para la clase de vegetales existieron 9 verdaderos negativos y 1 falso positivo, y con ello se obtienen los siguientes datos:

- Exactitud

$$Exactitud = 0.95 \Longrightarrow 95\%$$

- Tasa de error

$$Tasa\ de\ error = 0.05 \Longrightarrow 5\%$$

- Sensibilidad, exhaustividad

$$Sensibilidad = 1 \Longrightarrow 100\%$$

- Especificidad

$$Especificidad = 0.9 \Longrightarrow 90\%$$

- Precisión

$$Precisión = 0.91 \Longrightarrow 91\%$$

- Valor de predicción negativo

$$VPN = 1 \Longrightarrow 100\%$$

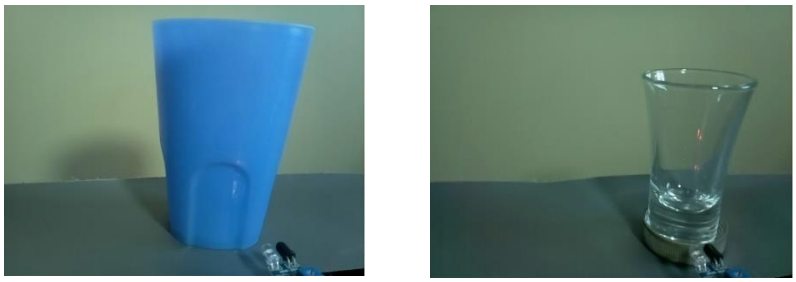
Prueba extra. Se intentó la clasificación de objetos con forma similar por lo que se empleó manzanas rojas y verdes, sin embargo, el servicio local no identificaba a esta fruta según su color, pues indistintamente del tipo la identificaba con la etiqueta de “manzana” únicamente, por tal motivo este grupo de clasificación será probado con el servicio local.

Pruebas de Textura. Para esto se establece 1 grupo de prueba, en él se establecen dos categorías, los materiales de vidrio y los materiales de plástico, empelando vasos y utensilios de tal material.

Prueba 3. Se emplean utensilios de plástico y vidrio como los mostrados en la figura 132.

Figura 132

Objetos de prueba 3.



Se realizan un total de 20 pruebas, 10 objetos de plástico y 10 objetos de vidrio, obteniendo el resultado presentado en la tabla 21.

Tabla 21

Resultado de prueba 3.

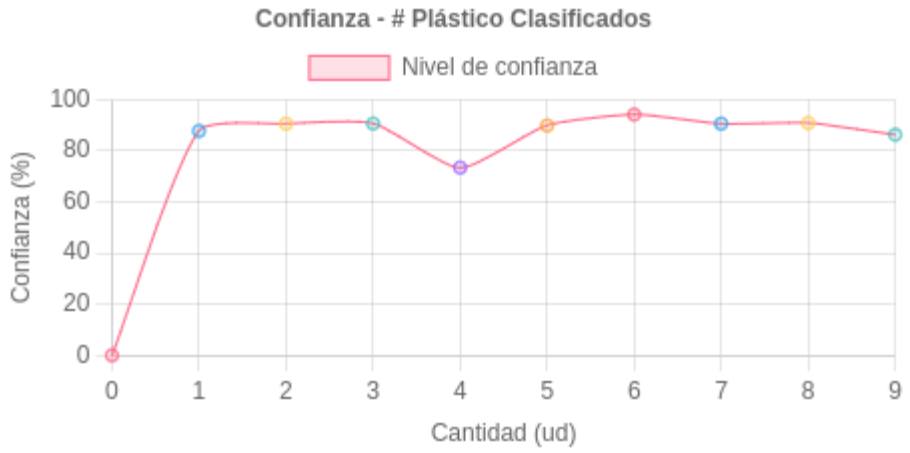
Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Plástico	***Vidrio***	***Plástico***	***Vidrio***	***Acierto [%]***	***Error [%]***
1	X		X		87.80	12.20
2	X		X		90.65	9.35
3	X		X		90.69	9.31
4	X			X	92.06	7.94
5	X		X		73.47	26.53
6	X		X		89.94	10.06
7	X		X		94.22	5.78
8	X		X		90.65	9.35
9	X		X		90.91	9.09

Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Plástico	***Vidrio***	***Plástico***	***Vidrio***	***Acierto [%]***	***Error [%]***
10	X		X		86.33	13.67
11		X		X	99.13	0.87
12		X		X	98.82	1.18
13		X	X		64.15	35.85
14		X		X	99.55	0.45
15		X		X	99.38	0.62
16		X		X	97.60	2.40
17		X		X	68.73	31.27
18		X		X	98.91	1.09
19		X		X	97.87	2.13
20		X		X	98.95	1.05

Para este grupo de prueba se generó un falso negativo del objeto 4, pues siendo de plástico fue identificado como vidrio, y como se muestra en la figura 133 solo se clasificaron 9 de los 10 objetos de plástico.

Figura 133

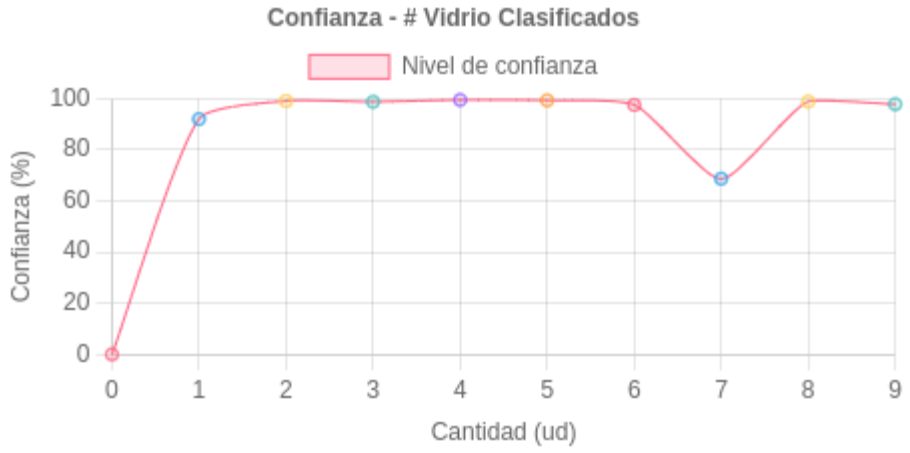
Confianza del objeto 1 de la prueba 3.



Para el caso de los objetos de vidrio existió un falso positivo, pues el objeto 13 fue identificado como plástico siendo de vidrio, por ello también solo se clasificaron 9 de 10 objetos de manera correcta como lo muestra la figura 134.

Figura 134

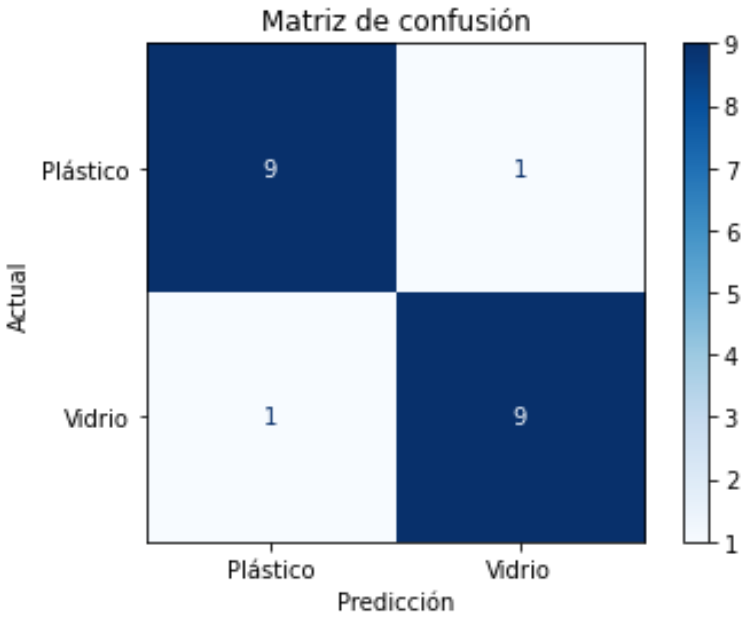
Confianza del objeto 2 de la prueba 3.



Para conseguir una mejor interpretación y evaluación del modelo de predicción para esta prueba de clasificación se genera una matriz de confusión representada en la figura 135.

Figura 135

Matriz de la tercera prueba.



En la matriz presentada se muestra un total de 20 imágenes de prueba con dos clases balanceadas, 10 imágenes para objetos de plástico y 10 imágenes para objetos de vidrio, para la clase de plástico existieron 9 verdaderos positivos y un falso negativo, mientras que para la clase de vidrio existieron 9 verdaderos negativos y 1 falso positivo, dando con ello lo siguiente:

- Exactitud

$$Exactitud = \frac{9+9}{20} = 0.9 \Longrightarrow 90\%$$

- Tasa de error

$$Tasa\ de\ error = \frac{2}{20} = 0.1 \Longrightarrow 10\%$$

- Sensibilidad

$$Sensibilidad = \frac{9}{1+9} = 0.9 \Longrightarrow 90\%$$

- Especificidad

$$Especificidad = \frac{9}{1+9} = 0.9 \implies 90\%$$

- Precisión

$$Precisión = \frac{9}{1+9} = 0.9 \implies 90\%$$

- Valor de predicción negativo

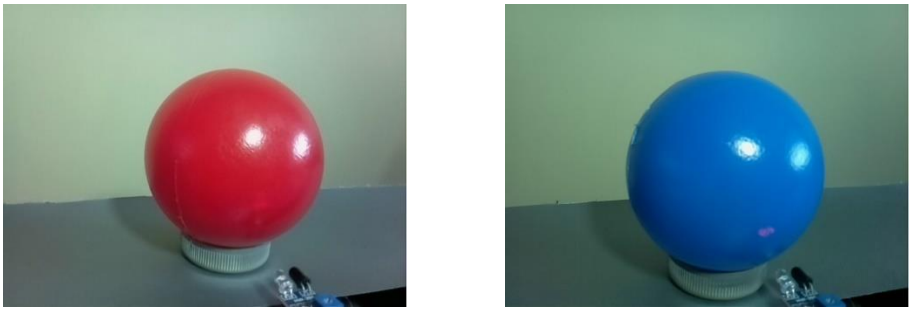
$$VPN = \frac{9}{1+9} = 0.9 \implies 90\%$$

Pruebas de Color. Para esto se establece 2 grupos de prueba, y con ellos 4 colores 2 encada prueba.

Prueba 4. Se emplean esferas de color rojo y azul como los mostrados en la figura 136.

Figura 136

Objetos de prueba 4.



Se realizan un total de 20 pruebas, 10 objetos de color rojo y 10 objetos de color azul, obteniendo el resultado presentado en la tabla 22.

Tabla 22

Resultados de la prueba 4.

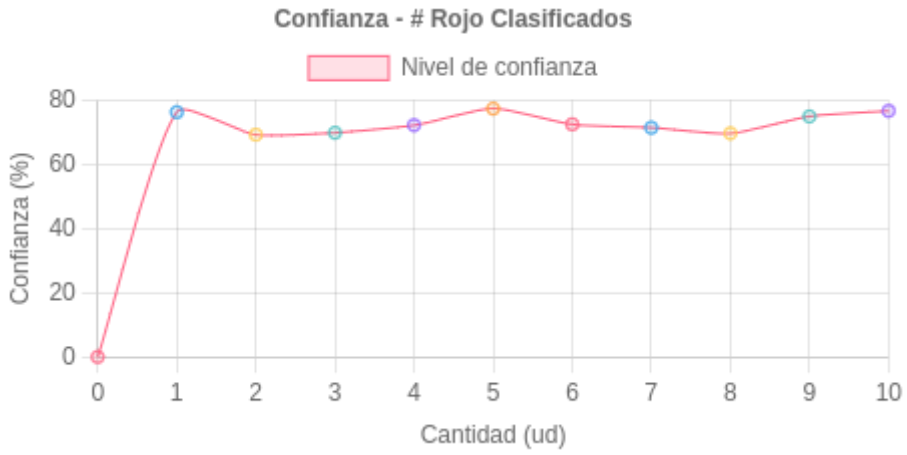
Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Rojo	***Azul***	***Rojo***	***Azul***	***Acierto [%]***	***Error [%]***
1	X		X		76.29	23.71
2	X		X		69.26	30.74
3	X		X		69.86	30.14
4	X		X		72.21	27.79
5	X		X		77.37	22.63
6	X		X		72.49	27.51
7	X		X		71.43	28.57
8	X		X		69.68	30.32
9	X		X		74.94	25.06
10	X		X		76.67	23.33
11		X		X	79.44	20.56
12		X		X	68.90	31.10
13		X		X	81.02	18.98
14		X		X	73.16	26.84
15		X		X	67.06	32.94
16		X		X	78.49	21.51
17		X		X	69.70	30.30
18		X		X	78.63	21.37
19		X		X	75.74	24.26

Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Rojo	***Azul***	***Rojo***	***Azul***	***Acierto [%]***	***Error [%]***
20		X		X	79.64	20.36

Para este grupo de prueba el porcentaje de acierto se vio reducido en comparación con los valores obtenidos en las pruebas realizadas en forma y textura, los valores se mantuvieron entre 69.26 % al 77.37 % como se muestra en la figura 137, sin embargo, se obtuvo una clasificación correcta de los 10 objetos rojos.

Figura 137

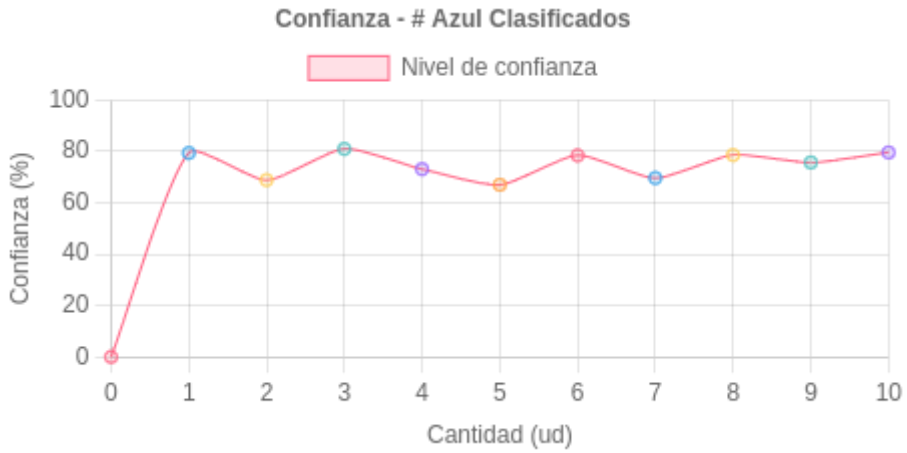
Confianza de objeto 1 de la prueba 4.



Para el caso de los objetos de color azul los resultados fueron similares, pues se obtuvieron 10 clasificaciones correctas y el porcentaje de acierto se mantuvo entre 67.06 % al 81.02 % como muestra la figura 138.

Figura 138

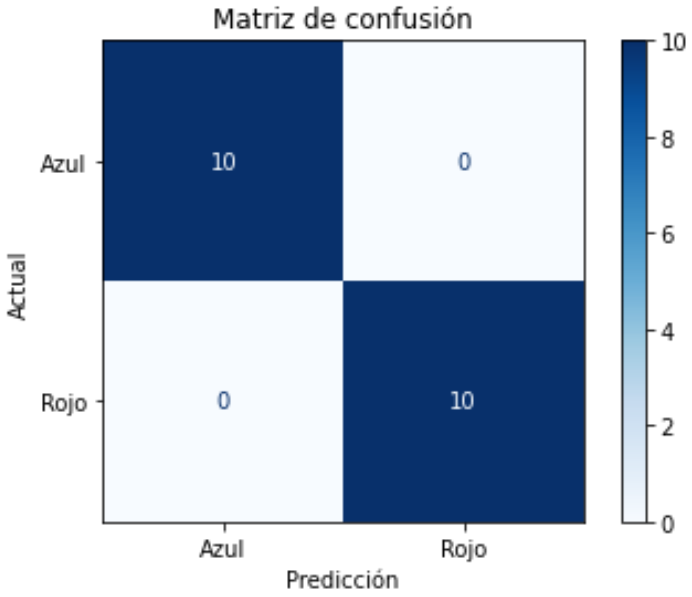
Confianza de objeto 2 de la prueba 4.



Para conseguir una mejor interpretación y evaluación del modelo de predicción para esta prueba de clasificación se genera una matriz de confusión que se lo puede observar en la figura 139.

Figura 139

Matriz de la prueba 4.



En la matriz presentada se muestra un total de 20 imágenes de prueba con dos clases balanceadas, 10 imágenes para objetos rojos y 10 imágenes para objetos azules, para la clase rojo existieron 10 verdaderos positivos, mientras que para la clase verde existieron 10 verdaderos negativos, obteniendo los siguientes resultados:

- Exactitud

$$Exactitud = \frac{10 + 10}{20} = 1 \Longrightarrow 100\%$$

- Tasa de error

$$Tasa\ de\ error = \frac{0}{20} = 0 \Longrightarrow 0\%$$

- Sensibilidad

$$Sensibilidad = \frac{10}{0 + 10} = 1 \Longrightarrow 100\%$$

- Especificidad

$$Especificidad = \frac{10}{0 + 10} = 1 \Longrightarrow 100\%$$

- Precisión

$$Precisión = \frac{10}{10 + 0} = 1 \Longrightarrow 100\%$$

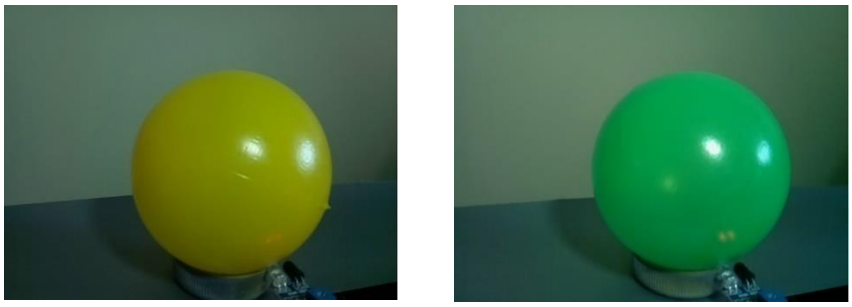
- Valor de predicción negativo

$$VPN = \frac{10}{10 + 0} = 1 \Longrightarrow 100\%$$

Prueba 5. Para la segunda prueba de color se emplean esferas de color amarillo y verde como los mostrados en la figura 140.

Figura 140

Objetos de prueba 5.



Se realizan un total de 20 pruebas, 10 objetos de color amarillo y 10 objetos de color verde, obteniendo el resultado presentado en la tabla 23.

Tabla 23

Resultado de prueba 5.

Num. de objeto	**Objeto ingresado**		**Predicción**		**Resultado de la predicción**	
	Amarillo	***Verde***	***Amarillo***	***Verde***	***Acierto [%]***	***Error [%]***
1	X		X		85.59	14.41
2	X		X		80.89	19.11
3	X		X		83.91	16.09
4	X		X		81.36	18.64
5	X		X		80.02	19.98
6	X		X		78.55	21.45
7	X		X		80.77	19.23
8	X		X		83.34	16.66
9	X		X		78.21	21.79

Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Amarillo	***Verde***	***Amarillo***	***Verde***	***Acierto [%]***	***Error [%]***
10	X		X		85.78	14.22
11		X		X	77.86	24.14
12		X		X	87.13	12.87
13		X		X	74.12	25.88
14		X		X	79.98	20.02
15		X		X	78.13	21.87
16		X		X	74.30	25.70
17		X		X	68.48	31.52
18		X		X	79.50	20.50
19		X		X	78.99	21.01
20		X		X	69.82	30.18

El resultado obtenido es similar al grupo en la prueba 1, pues se obtuvieron 10 clasificaciones correctas para los objetos de color amarillo, y mantuvieron un porcentaje de acierto de entre 78.21 % al 85.78 % mostrado en la figura 141.

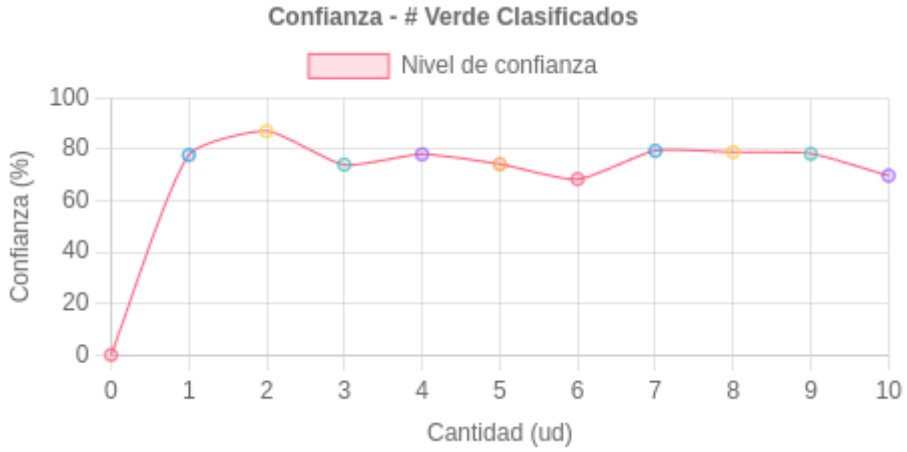
Figura 141

Confianza de objeto 1 de la prueba 5.

Para el caso de los objetos de color verde los resultados fueron similares, pues se obtuvieron 10 clasificaciones correctas y el porcentaje de acierto se mantuvo entre 68.48 % al 79.98 % como muestra la figura 142.

Figura 142

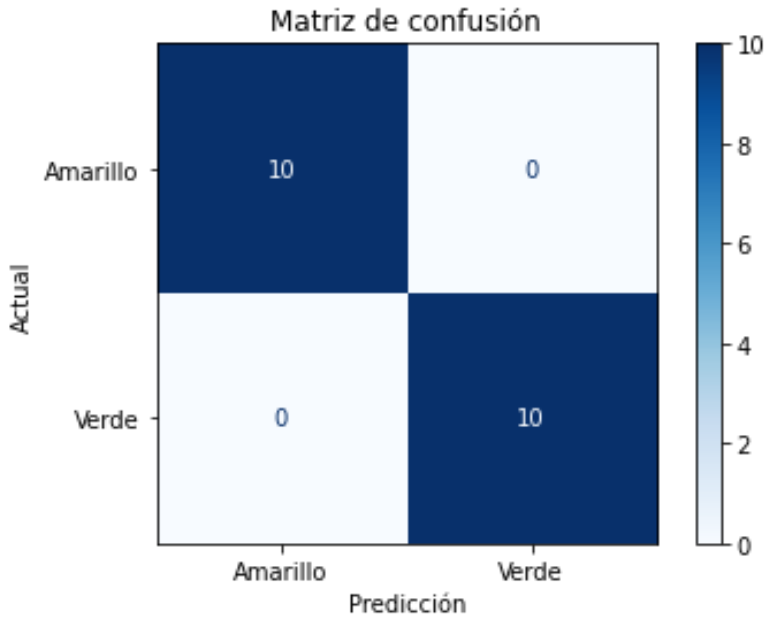
Confianza del objeto 2 de la prueba 5.



Para conseguir una mejor interpretación y evaluación del modelo de predicción para esta prueba de clasificación se genera una matriz de confusión mostrado en la figura 143.

Figura 143

Matriz de la prueba 5.



En la matriz presentada se muestra un total de 20 imágenes de prueba con dos clases balanceadas, 10 imágenes para objetos amarillos y 10 imágenes para objetos verde, para la clase amarillo existieron 10 verdaderos positivos, mientras que para la clase verde existieron 10 verdaderos negativos, obteniendo los siguientes resultados:

- Exactitud

$$Exactitud = 1 \Longrightarrow 100\%$$

- Tasa de error

$$Tasa\ de\ error = 0 \Longrightarrow 0\%$$

- Sensibilidad

$$Sensibilidad = 1 \Longrightarrow 100\%$$

- Especificidad

$$Especificidad = 1 \Longrightarrow 100\%$$

- Precisión

$$Precisión = 1 \implies 100\%$$

- Valor de predicción negativo

$$VPN = 1 \implies 100\%$$

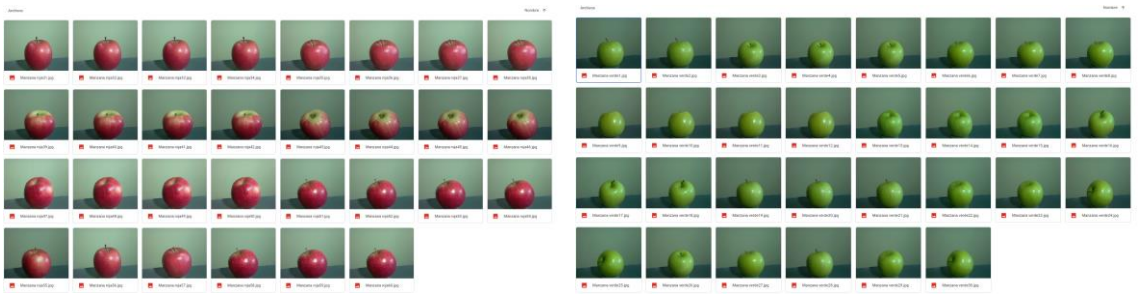
Pruebas del Servicio Local

El propósito del servicio local es cubrir la necesidad de clasificar elementos que el servicio en la nube no puede identificar o lo haga de manera incorrecta. Durante el proceso de pruebas del servicio en la nube se identificaron ciertos objetos que cumplen con la característica antes mencionada y se realizaron las pruebas con ellos.

Prueba 1. Durante las pruebas de forma, se identificó que el grupo de manzanas era identificado siempre solo como manzana, indistintamente si es roja o verde, por ello la primera prueba consta de estos dos grupos, con manzanas rojas y verdes. El servicio local requiere de un data set de ambos objetos, en la figura 144 se ilustra tal grupo de datos empleado para el entrenamiento del modelo.

Figura 144

Lista de objetos para la prueba 1.



Una vez listo el modelo, se carga al sistema y se emplearon frutas diferentes a las empleadas para el entrenamiento como las mostradas en la figura 145.

Figura 145

Objetos para prueba local 1.

Se realizan un total de 20 pruebas, 10 manzanas rojas y 10 manzanas verdes, obteniendo el resultado presentado en la tabla 24.

Tabla 24

Resultados de la prueba local 1.

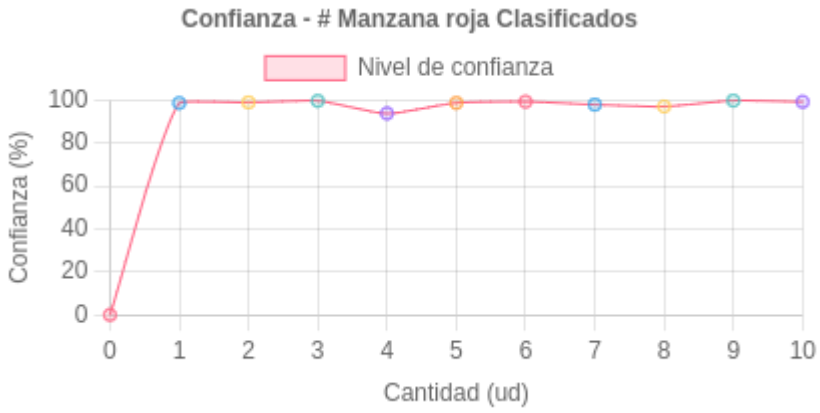
Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Manzana roja	***Manzana verde***	***Manzana roja***	***Manzana verde***	***Acierto [%]***	***Error [%]***
1	X		X		98.69	1.31
2	X		X		99.01	0.99
3	X		X		99.72	0.28
4	X		X		93.88	6.12
5	X		X		98.77	1.23
6	X		X		97.98	2.02
7	X		X		97.11	2.89
8	X		X		99.83	0.17
9	X		X		99.27	0.73

Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Manzana roja	***Manzana verde***	***Manzana roja***	***Manzana verde***	***Acierto [%]***	***Error [%]***
10	X		X		98.69	1.31
11		X		X	92.94	7.06
12		X		X	97.01	2.99
13		X		X	91.48	8.52
14		X	X		62.93	37.07
15		X		X	96.60	3.40
16		X		X	88.73	11.27
17		X		X	91.30	8.70
18		X		X	55.24	44.76
19		X		X	89.22	10.78
20		X		X	93.95	6.05

Para el primero grupo de manzanas rojas, se clasificaron las 10 de manera correcta con un porcentaje de acierto alto de entre 97.1 % al 99.83 como se ilustra en la figura 146.

Figura 146

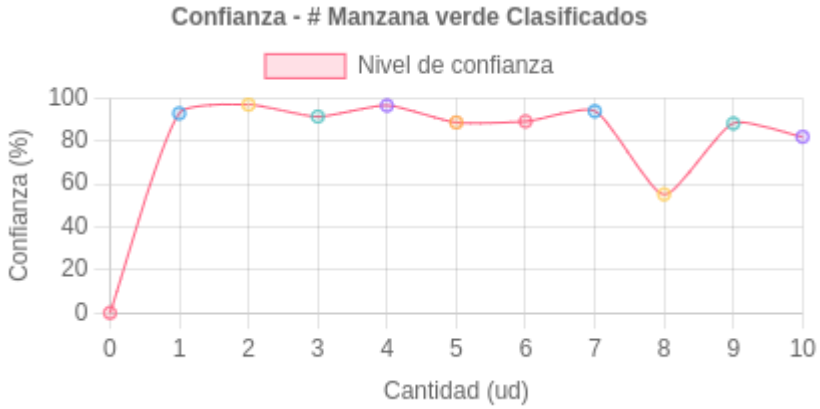
Confianza del objeto 1 de la prueba local 1.



Para el caso de las manzanas verdes se obtuvieron resultados presentados en la figura 147 en las que existió un falso negativo para el objeto número 14, y 9 manzanas verdes clasificadas de manera correcta.

Figura 147

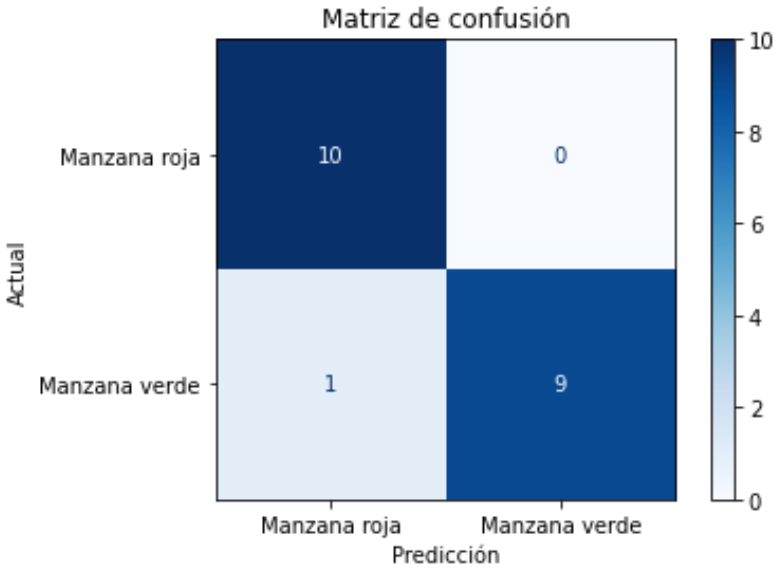
Confianza del objeto 2 de la prueba local 1.



Para conseguir una mejor interpretación y evaluación del modelo de predicción para esta prueba de clasificación se genera una matriz de confusión como se ve en la figura 148.

Figura 148

Matriz de la prueba local 1.



En la matriz presentada se muestra un total de 20 imágenes de prueba con dos clases balanceadas, 10 imágenes para manzanas rojas y 10 imágenes para manzanas verdes, para la clase de manzanas existieron 10 verdaderos positivos, mientras que para la clase de peras existieron 9 verdaderos negativos y 1 falso positivo.

- Exactitud

$$Exactitud = \frac{10 + 9}{20} = 0.95 \Longrightarrow 95\%$$

- Tasa de error

$$Tasa\ de\ error = \frac{1}{20} = 0.05\ \Longrightarrow 5\%$$

- Sensibilidad, exhaustividad

$$Sensibilidad = \frac{10}{0 + 10} = 1\ \Longrightarrow 100\%$$

- Especificidad

$$Especificidad = \frac{9}{1+9} = 0.9 \implies 90\%$$

- Precisión

$$Precisión = \frac{10}{1+10} = 0.91 \implies 91\%$$

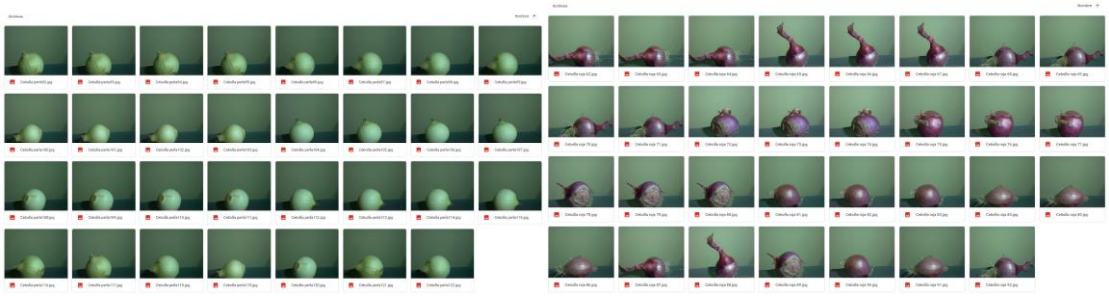
- Valor de predicción negativo

$$VPN = \frac{9}{9} = 1 \implies 100\%$$

Prueba 2. Otro grupo identificado que no pudo ser clasificado por el servicio en la nube fueron las cebollas perla y cebollas roja, pues ambas eran identificadas solamente como cebollas, la figura 149 ilustra el data set de ambos objetos empleado para el entrenamiento del modelo.

Figura 149

Lista de objetos de la prueba local 2.



Una vez listo el modelo, se carga al sistema y se emplearon diferentes cebollas como las mostradas en la figura 150 para las pruebas.

Figura 150

Objetos de la prueba local 2.

Se realizan un total de 20 pruebas, 10 cebollas perlas y 10 cebollas rojas, obteniendo el resultado presentado en la tabla 25.

Tabla 25

Resultados de la prueba local 2.

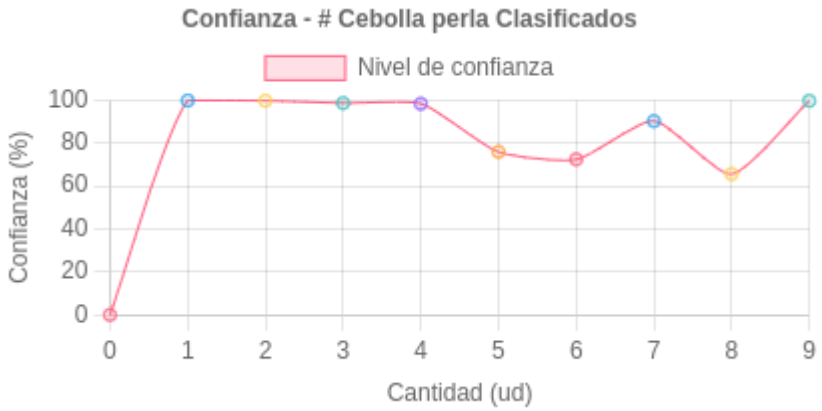
Num. de objeto	**Objeto ingresado**		**Predicción**		**Resultado de la predicción**	
	Cebolla perla	***Cebolla roja***	***Cebolla perla***	***Cebolla roja***	***Acierto [%]***	***Error [%]***
1	X		X		99.88	0.12
2	X		X		99.77	0.23
3	X		X		98.69	1.31
4	X		X		98.40	1.60
5	X		X		98.40	1.60
6	X		X		72.46	27.54
7	X		X		90.27	9.73
8	X		X		96.44	3.56
9	X			X	65.53	34.74

Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Cebolla perla	***Cebolla roja***	***Cebolla perla***	***Cebolla roja***	***Acierto [%]***	***Error [%]***
10	X		X		99.74	0.26
11		X		X	100	0
12		X		X	100	0
13		X		X	100	0
14		X		X	100	0
15		X		X	100	0
16		X		X	100	0
17		X		X	100	0
18		X		X	100	0
19		X		X	100	0
20		X		X	100	0

Para el primero grupo de cebollas perla, se clasificaron 9 de manera correcta y el objeto 9 correspondiente a una cebolla perla fue identificada como roja, el nivel de confianza o acierto se mantuvo entre 72.46 % al 99.88 % como se ilustra en la figura 151.

Figura 151

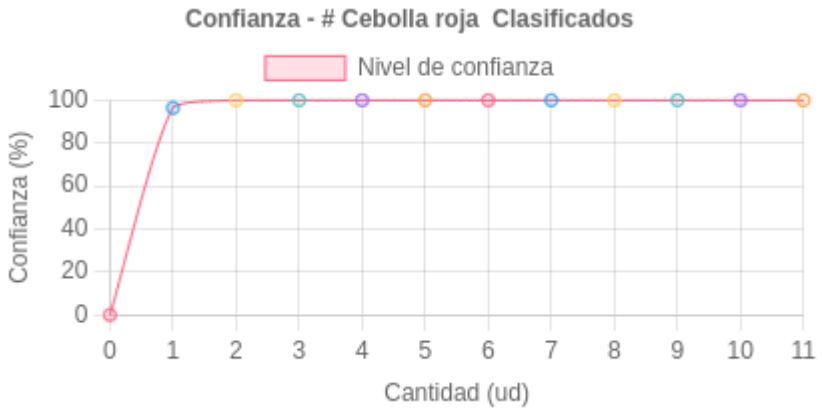
Confianza del objeto 1 de la prueba local 1.



Para el caso de las cebollas rojas se da una clasificación de los 10 objetos de manera correcta, y con la peculiaridad de mantener su nivel de confianza o acierto al 100 por ciento, como se ilustra en la figura 152.

Figura 152

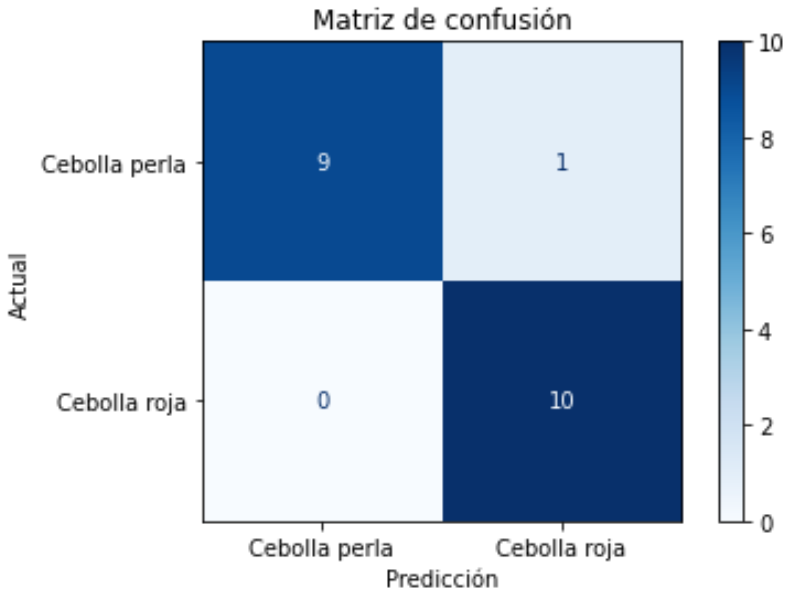
Confianza del objeto 2 de la prueba local 2.



Para conseguir una mejor interpretación y evaluación del modelo de predicción para esta prueba de clasificación se genera una matriz de confusión como se ve en la figura 153.

Figura 153

Matriz de la prueba local 2.



En la matriz presentada se muestra un total de 20 imágenes de prueba con dos clases balanceadas, 10 imágenes para cebollas perla y 10 imágenes para cebollas rojas, para la clase de cebolla perla existieron 9 verdaderos positivos un falso negativo, mientras que para la clase de cebollas rojas existieron 10 verdaderos negativos, obteniendo los siguientes resultados:

- Exactitud

$$Exactitud = \frac{9 + 10}{20} = 0.95 \Longrightarrow 95\%$$

- Tasa de error

$$Tasa\ de\ error = \frac{1}{20} = 0.05\ \Longrightarrow 5\%$$

- Sensibilidad, exhaustividad

$$Sensibilidad = \frac{9}{1 + 9} = 0.9\ \Longrightarrow 90\%$$

- Especificidad

$$Especificidad = \frac{10}{0+10} = 1 \implies 100\%$$

- Precisión

$$Precisión = \frac{9}{9+0} = 1 \implies 100\%$$

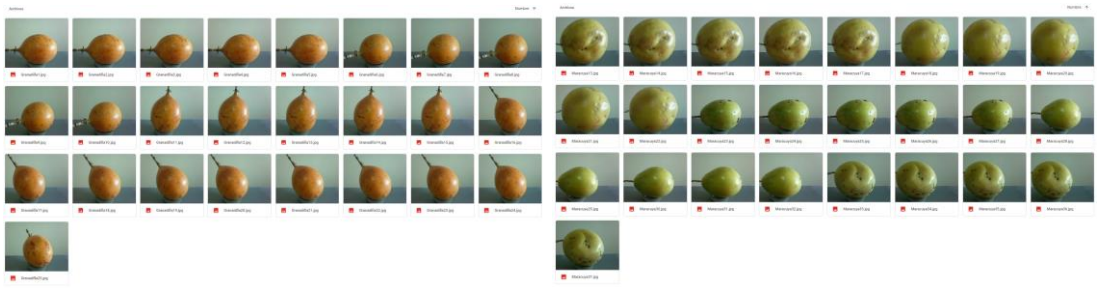
- Valor de predicción negativo

$$VPN = \frac{10}{1+10} = 0.91 \implies 91\%$$

Prueba 3. Finalmente, otro grupo identificado que no pudo ser clasificado por el servicio en la nube fueron la granadilla con el maracuyá, pues ambas eran identificadas solamente como maracuyá, la figura 154 ilustra el data set de ambos objetos empleado para el entrenamiento del modelo.

Figura 154

Lista de objetos de la prueba local 3.



Una vez listo el modelo, se carga al sistema y se emplearon diferentes cebollas como las mostradas en la figura 155 para las pruebas.

Figura 155

Objetos de la prueba local 3.

Se realizan un total de 20 pruebas, 10 de granadilla y 10 de maracuyá, obteniendo el resultado presentado en la tabla 26.

Tabla 26

Resultados de la prueba local 3.

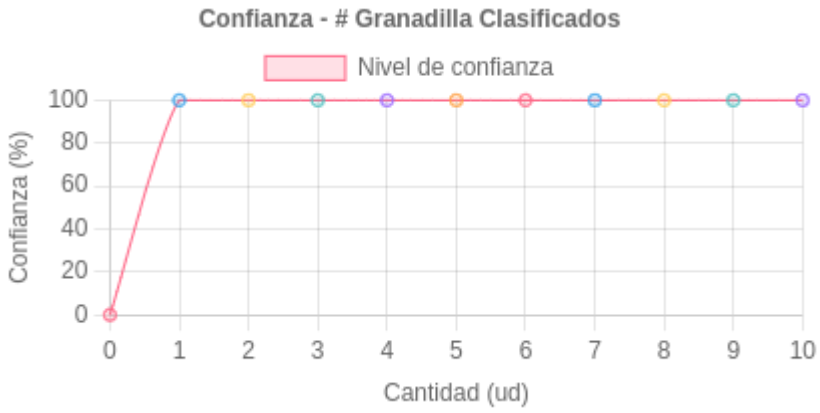
Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Granadilla	***Maracuyá***	***Granadilla***	***Maracuyá***	***Acierto [%]***	***Error [%]***
1	X		X		100	0
2	X		X		100	0
3	X		X		99.99	0.01
4	X		X		100	0
5	X		X		100	0
6	X		X		100	0
7	X		X		100	0
8	X		X		100	0
9	X		X		100	0

Num. de objeto	Objeto ingresado		Predicción		Resultado de la predicción	
	Granadilla	***Maracuyá***	***Granadilla***	***Maracuyá***	***Acierto [%]***	***Error [%]***
10	X		X		100	0
11		X		X	100	0
12		X	X		91.62	8.38
13		X		X	99.50	0.50
14		X		X	98.57	1.43
15		X		X	99.97	0.03
16		X		X	99.96	0.04
17		X		X	99.97	0.03
18		X		X	99.98	0.02
19		X		X	98.90	1.10
20		X		X	96.98	3.02

Para el primero grupo de granadilla, se clasificaron 10 de manera correcta, el nivel de confianza o acierto se mantuvo al 100 % como se ilustra en la figura 156.

Figura 156

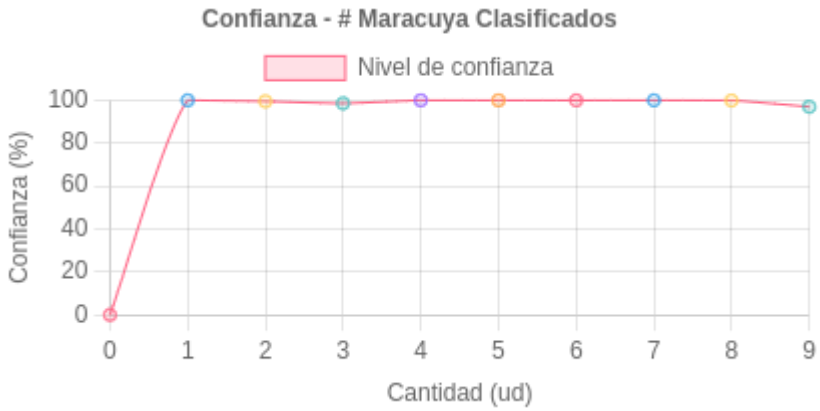
Confianza del objeto 1 de la prueba local 3.



Para el caso del maracuyá se da una clasificación de 9 objetos de manera correcta y 1 de manera incorrecta y manteniendo el nivel de confianza de entre 91.62 % al 100 % como se ilustra en la figura 157.

Figura 157

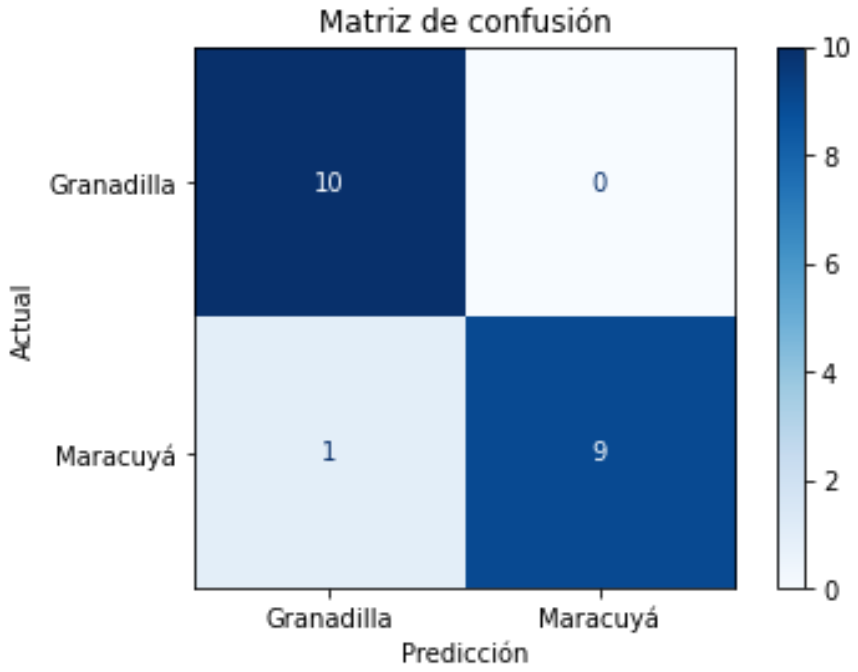
Confianza del objeto 2 de la prueba local 3.



Para conseguir una mejor interpretación y evaluación del modelo de predicción para esta prueba de clasificación se genera una matriz de confusión como se ve en la figura 158.

Figura 158

Matriz de la prueba local 3.



En la matriz presentada se muestra un total de 20 imágenes de prueba con dos clases balanceadas, 10 imágenes para granadilla y 10 imágenes para maracuyá, para la clase de granadilla existieron 9 verdaderos positivos y 1 falso negativo, mientras que para la clase de maracuyá existieron 10 verdaderos negativos, obteniendo los siguientes resultados:

- Exactitud

$$Exactitud = \frac{10 + 9}{20} = 0.95 \Longrightarrow 95\%$$

- Tasa de error

$$Tasa\ de\ error = \frac{1}{20} = 0.05 \ \Longrightarrow 5\%$$

- Sensibilidad, exhaustividad

$$Sensibilidad = \frac{10}{10 + 0} = 1 \ \Longrightarrow 100\%$$

- Especificidad

$$Especificidad = \frac{9}{1+9} = 0.9 \implies 90\%$$

- Precisión

$$Precisión = \frac{10}{10+1} = 0.91 \implies 91\%$$

- Valor de predicción negativo

$$VPN = \frac{9}{9} = 1 \implies 100\%$$

Una vez realizado las pruebas de funcionamiento y haber obtenido los resultados del desempeño, se resumen estos datos en la tabla 27, en la que se detallan los valores para cada prueba del servicio en la nube.

Tabla 27

Resumen de resultados del servicio en la nube.

		Exactitud [%]	**Tasa de error [%]**	**Sensibilidad [%]**	**Especificidad [%]**	**Precisión [%]**	**VPN [%]**	**Valor F [%]**
Forma	***Prueba 1***	95	5	100	90	91	100	95
	Prueba 2	95	5	100	90	91	100	95
Textura	***Prueba 3***	90	10	90	90	90	90	90
	Prueba 4	100	0	100	100	100	100	100
Color	***Prueba 5***	100	0	100	100	100	100	100

Para las pruebas realizadas con la forma de los objetos con los grupos de manzana-pera y de frutas-verduras el porcentaje de exactitud se mantuvo en un 95% y el valor F también en 95% determinando así que el desempeño del modelo de clasificación funciona de manera correcta para objetos con características similares a las presentadas en las pruebas.

Para la prueba realizada con la textura de plástico-vidrio se obtuvo un valor de exactitud del 90 % y un valor F de 90 % concluyendo con que el modelo de clasificación presentado por el servicio en la nube posee un desempeño que se considera adecuado para el sistema de clasificación.

Finalmente, para las pruebas de color realizadas para los grupos de rojo-azul y amarillo-verde se consiguieron valores de exactitud y valor F de un 100% determinando que el modelo de clasificación del servicio en la nube funciona de manera correcta para identificar colores.

De la misma manera se resume y detalla dentro de la tabla 28 los valores del desempeño del modelo para cada una de las pruebas realizadas al servicio local.

Tabla 28

Resumen de resultados del servicio en la local.

	Exactitud [%]	**Tasa de error [%]**	**Sensibilidad [%]**	**Especificidad [%]**	**Precisión [%]**	**VPN [%]**	**Valor F [%]**
Prueba 1	95	5	100	90	91	100	95
Prueba 2	95	5	90	100	100	91	95
Prueba 3	95	5	100	90	91	100	95

Para las pruebas realizadas al servicio local, se generaron 3 grupos distintos de manzana roja-manzana verde, cebolla perla- cebolla roja y granadilla-maracuyá en los cuales el

valor de la exactitud se mantuvo en un 95 % y el valor F también en un 95 %, solo existió variaciones entre los valores de sensibilidad y precisión, pero dado que en el sistema de clasificación la prioridad de clasificación del objeto uno con respecto al objeto dos no es relevante solo se mantiene en análisis el valor F y exactitud concluyendo que los modelos de clasificación generados para las tres pruebas funcionan de manera correcta en el sistema.

Interfaz

Interfaz local

Interfaz servicio en la nube. La figura 159 muestra la imagen capturada, en la parte derecha la predicción junto con la posición del objeto, e inferior al bloque de predicción se observa el estado de los diferentes actuadores, y en la figura 160 se pude observar los elementos a clasificar y los botones de control del sistema.

Figura 159

Bloque inicial de la página del servicio en la nube.

Figura 160

Bloque de control de la página del servicio en la nube.

En la parte inferior al bloque anterior se encuentra el video del proceso en tiempo real, y a la derecha, la predicción del objeto junto con su imagen, una gráfica de dona indicando el valor de acierto y error de la predicción, y los contadores de cada objeto cómo se lo aprecia en la figura 161. Al final de la página se logra ver las gráficas de desempeño de cada objeto como se muestra en la figura 162.

Figura 161

Bloque de video y resultados de la página del servicio en la nube.

Figura 162

Bloque de análisis de la página del servicio en la nube.

Interfaz servicio local. En la figura 163 se ilustra primero el bloque de Creación de Repositorio en la cual se selecciona la posición del objeto y el elemento que se desea usar, en el bloque de Avisos observar información sobre los datos, en la parte derecha se capturan las imágenes con las cuales va a ser entrenada la red. Y en la figura 164 se pude observar los nombres de los elementos previamente entrenados, y los botones de control del sistema.

Figura 163

Bloque inicial de la página del servicio en local.

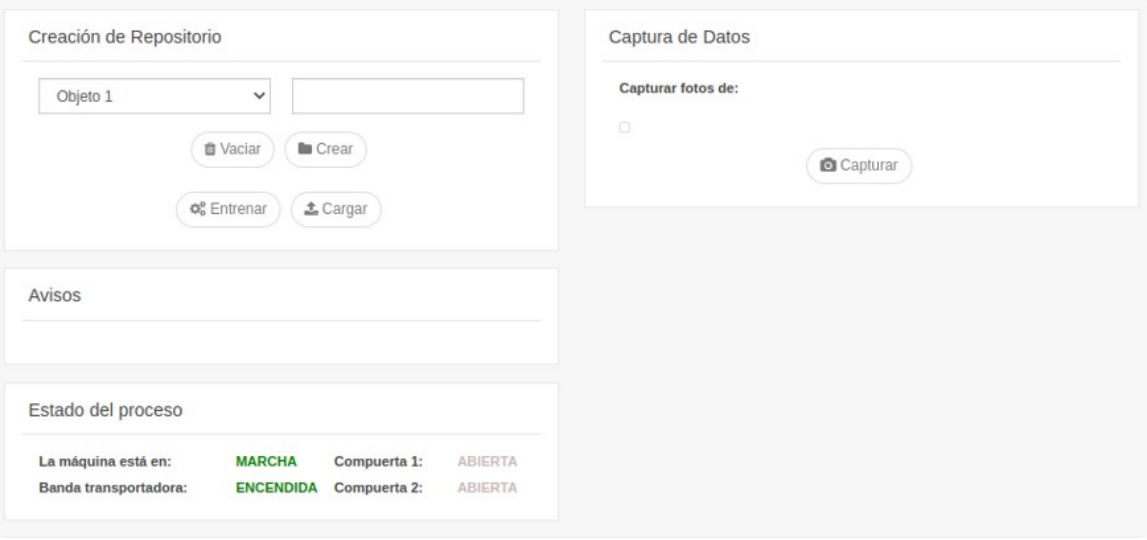
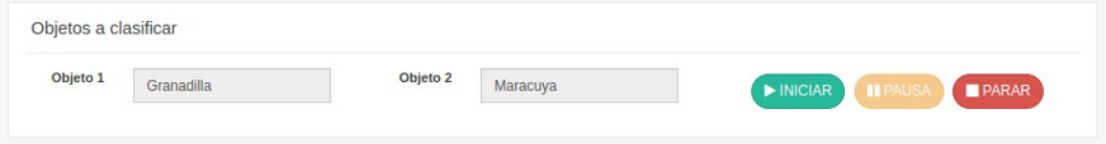


Figura 164

Bloque de control de la página del servicio en local.



En la parte inferior al bloque anterior se encuentra el video del proceso en tiempo real, y a la derecha, la predicción del objeto junto con su imagen, una gráfica de dona indicando el valor de acierto y error de la predicción, y los contadores de cada objeto cómo se lo aprecia en la figura 165. Al final de la página se logra ver las gráficas de desempeño de cada objeto como se lo observa en la figura 166.

Figura 165

Bloque de video y resultados de la página del servicio en local.

Figura 166

Bloque análisis de la página del servicio en local.

Interfaz Remota. Las interfaces de los datos tanto como el de servicio en la Nube y el servicio local cuentan con la misma apariencia, la diferencia se encuentra en que cada página se conecta con un diferente enlace según el servicio que se esté usando en el sistema.

En la figura 167 se puede observar los objetos que van a ser clasificados, y los diferentes estados de los actuadores del sistema, y la figura 168 muestra la predicción del objeto con su imagen y una gráfica de donas con el resultado de acierto y error.

Figura 167

Bloque inicial de la página remota.

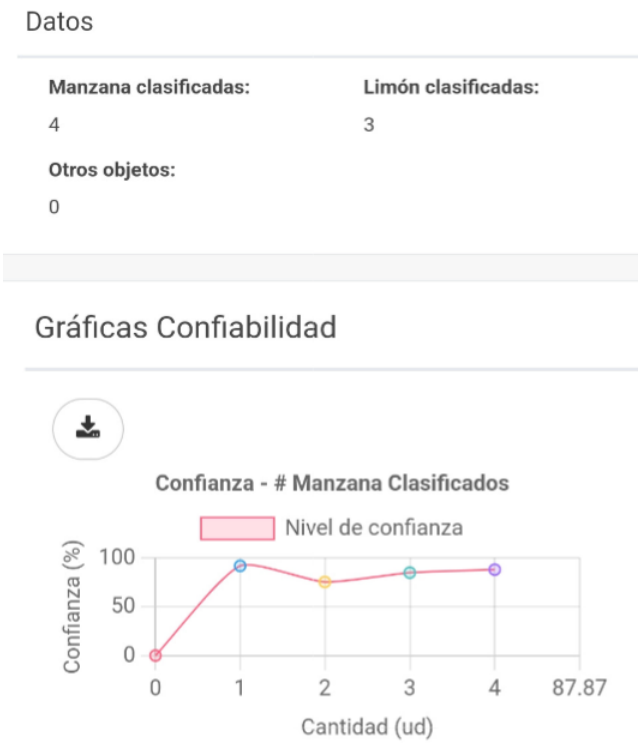
Figura 168

Bloque de resultados de la página remota.

En la figura 169 se observa los contadores de los objetos y las gráficas de desempeño en cada predicción.

Figura 169

Bloque de análisis de la página remota.



Pruebas de carga

Es importante conocer que la arquitectura de software se encuentra en buen estado, por lo que una vez el proyecto se encuentre completado y despegado en la web, se debe realizar pruebas de funcionamiento del servidor que aloja la página publica, esto se lo realiza mediante un script de Gatling basado en lenguaje Scala, se tiene que establecer las peticiones que son posibles de enviar, debido a que es una página de supervisión no cuenta con diversas funcionalidades, por lo que para este caso existen dos peticiones, que representan el número de botones que pueden realizar peticiones.

El código Gatling está conformado por tres partes principales, en la primera se define el tipo de protocolo, para este caso se usó el protocolo HTTP y sus atributos importantes como se lo muestra en la figura 170.

Figura 170

Configuración de protocolo.

```
//Protocolo
 val httpProtocol = http
  .baseUrl("https://sistemaclasificador.herokuapp.com")
  .header(name="Accept", value="application/json")
  .header(name="content-type", value="application/json")
```

En la segunda parte se debe crear el escenario, por lo tanto, se debe especificar el tipo y las peticiones que se van a realizar, y a cada una se le debe asignar un nombre, como se ilustra en la figura 171.

Figura 171

Creación de escenario.

```
//Escenario
val scn = scenario("Simulation1")
 .exec(
  http("Prueba Local")
  .get("/Local/"))

 .exec(
  http("Prueba Nube")
  .get("/Nube/"))
```

Finalmente, en la figura 172, se muestra el código que tiene la configuración final para la ejecución, en el cual se define el número de usuarios que se pretende simular, el tiempo en el que se debe ejecutar las sentencias y el protocolo que se usará para dichas pruebas, para esto se pretende simular un ingreso ascendente de usuarios, hasta llegar al número indicado, de esta forma se puede representar un escenario real.

Figura 172

Configuración de ingreso de usuarios.

```
setUp(scn.inject(rampUsers(4000).during(90))).protocols(httpProtocol)
```

En la figura 173, se puede notar la respuesta del servidor al ingresar 4000 usuarios, al ser demasiados no se han realizado todas las peticiones con éxito, el 6% ha fallado, en cuanto el restante 94% se distribuye de la siguiente forma: el 1% se logró enviar en un tiempo menor a 800 ms, el otro 1% se lo realizó entre 800 ms y 1200 ms, y el resto fue en el tiempo mayor a 1200 ms. En la figura 174 se puede observar detalladamente las peticiones realizadas a la página web, y en la figura 175, se muestra el número máximo de peticiones por segundo realizado junto con los usuarios activos en ese momento.

Figura 173

Informe de peticiones recibidas.

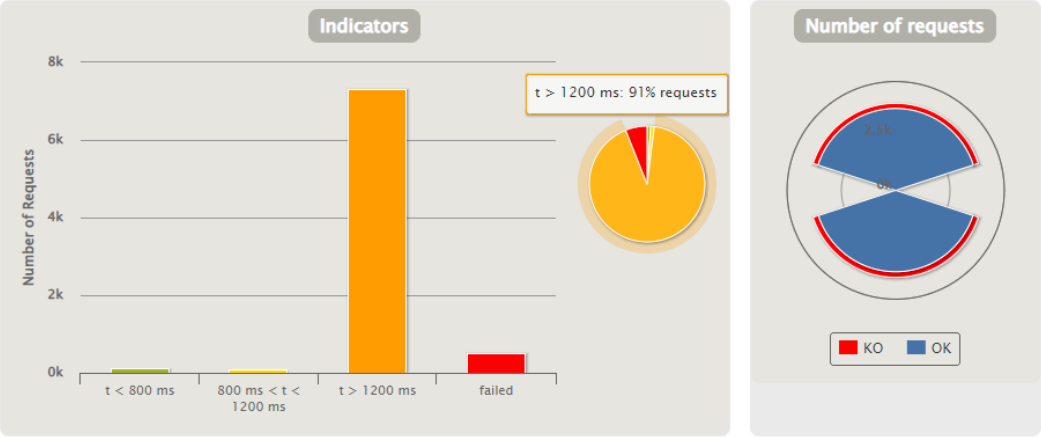


Figura 174

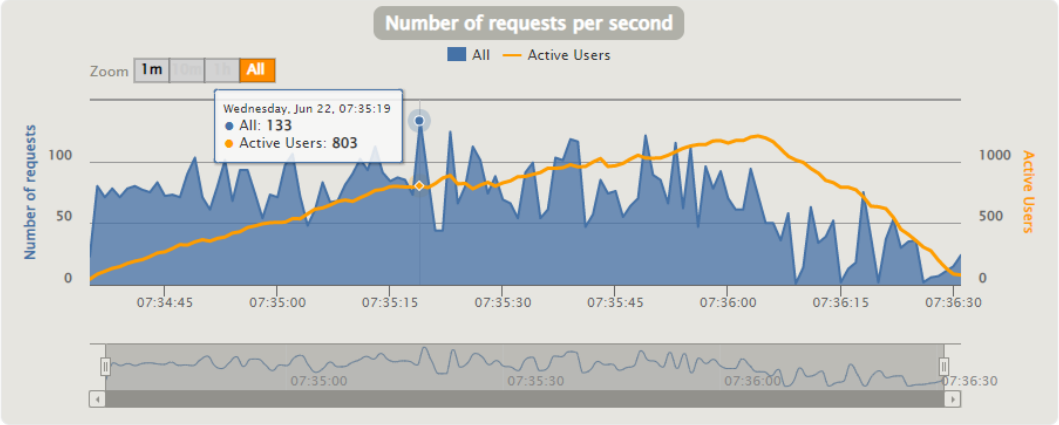
Información de las peticiones realizadas.

STATISTICS — Expand all groups | Collapse all groups

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	8000	7497	503	6%	68.376	151	6580	11674	28686	50833	60038	9873	9426
Prueba Local	4000	3759	241	6%	34.188	399	7627	15051	30913	47866	60016	11312	9832
Prueba Nube	4000	3738	262	7%	34.188	151	5858	9268	24421	51083	60038	8434	8769

Figura 175

Número de peticiones máximas.



Pruebas de carga con 100 usuarios.

En la figura 176, se puede observar que se han realizado con éxito las 100 peticiones, y dentro de estas el 92% de ellas se realizaron en un tiempo menor a 800 ms, y el 12% restante se lo realizó en un tiempo mayor a 1200 ms, y en la figura 177, se puede ver la cantidad máxima de peticiones realizadas por segundo, junto con la representación de usuarios activos.

Figura 176

Informe de peticiones recibidas con 100 usuarios.

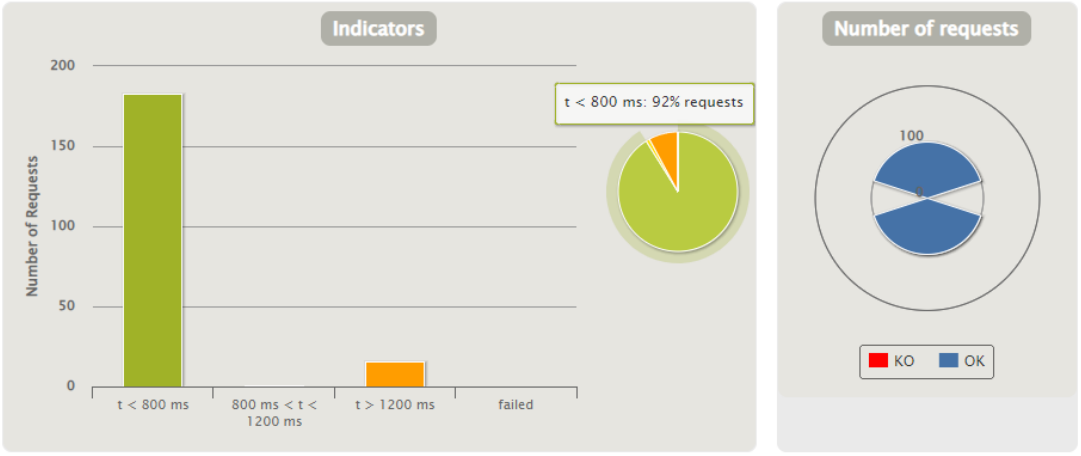
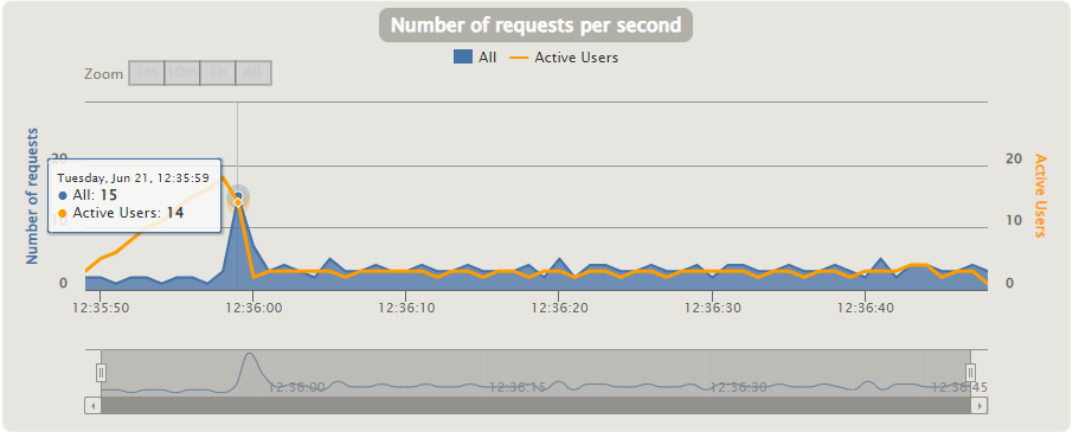


Figura 177

Número de peticiones por segundo con 100 usuarios.



Pruebas de carga con 200 usuarios.

En la figura 178, se puede observar que se han realizado con éxito las 200 peticiones, y dentro de estas el 100% de ellas se realizaron en un tiempo menor a 800 ms, y en la figura 179, se puede ver la cantidad máxima de peticiones realizadas por segundo, junto con la representación de usuarios activos.

Figura 178

Informe de peticiones recibidas con 200 usuarios.

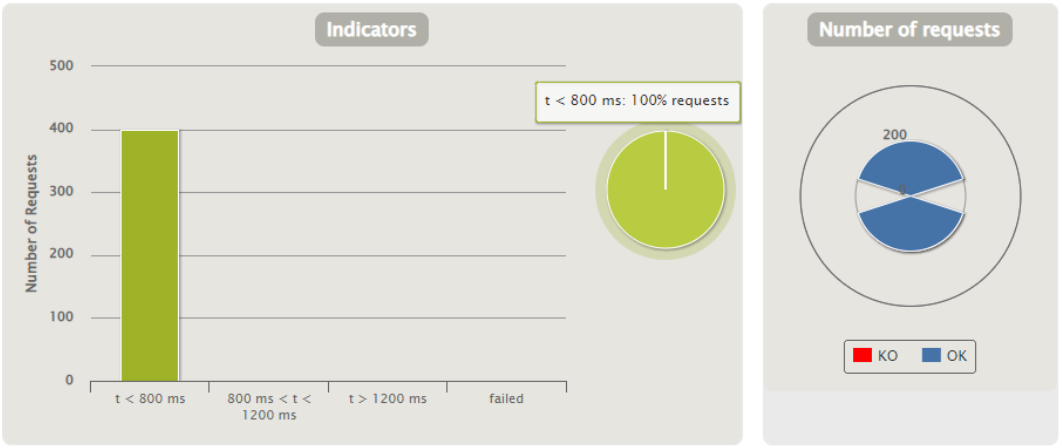
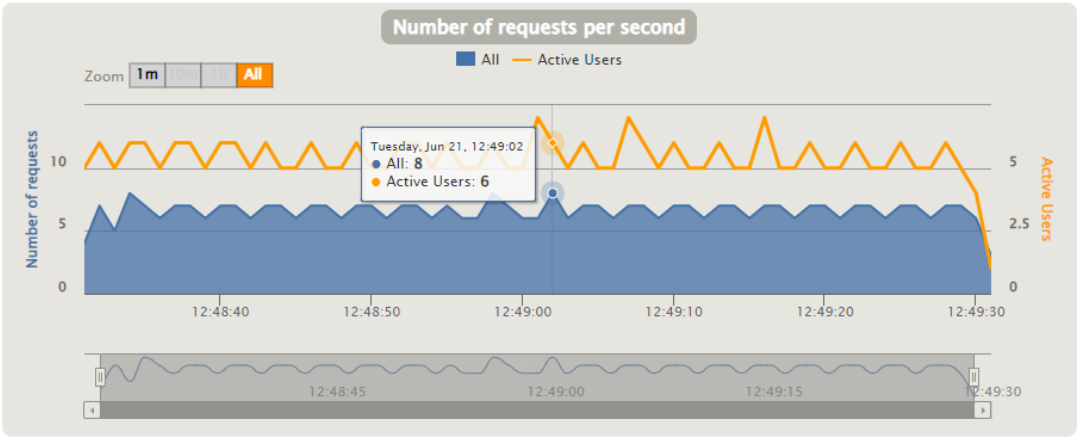


Figura 179

Número de peticiones por segundo con 200 usuarios.



Pruebas de carga con 500 usuarios.

En la figura 180, se puede observar que se han realizado con éxito las 500 peticiones, y dentro de estas el 99% de ellas se realizaron en un tiempo menor a 800 ms, y el 1% restante se lo realizó en un tiempo mayor a 1200 ms, y en la figura 181, se puede ver la cantidad máxima de peticiones realizadas por segundo, junto con la representación de usuarios activos.

Figura 180

Informe de peticiones recibidas con 500 usuarios.

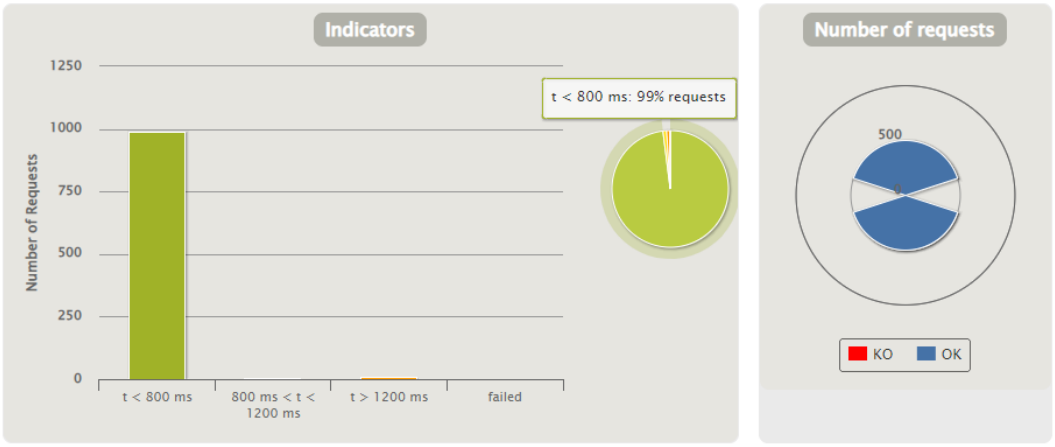
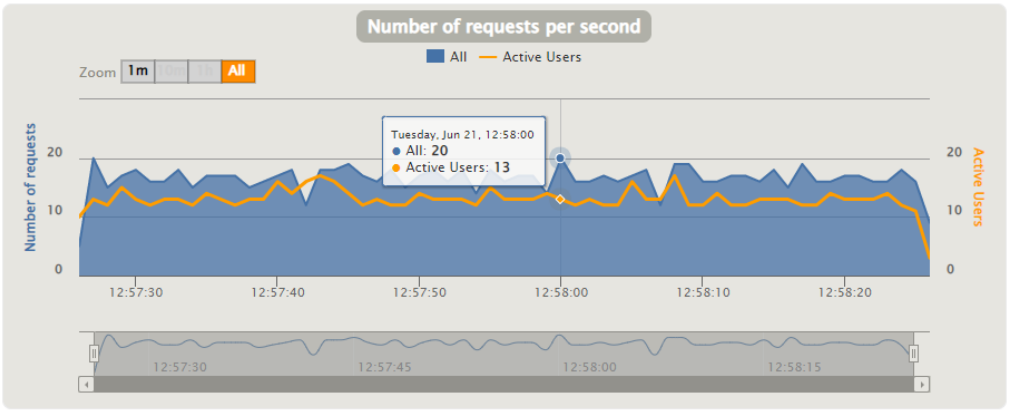


Figura 181

Informe de peticiones recibidas con 500 usuarios.



Pruebas de carga con 1000 usuarios.

En la figura 182, se puede observar que se han realizado las 1000 peticiones, y en este punto ya se pude observar una diferencia entre las anteriores, el 49% de las peticiones realizadas se lo hizo en un tiempo menor a 800 ms, el 10% lo realizó en un tiempo mayor a 800 ms y menor a 1200 ms, un 39% fue mayor a 1200 ms, y el 1% restante fueron fallos, y en la

figura 183, se puede ver la cantidad máxima de peticiones realizadas por segundo, junto con la representación de usuarios activos.

Figura 182

Informe de peticiones recibidas con 1000 usuarios.

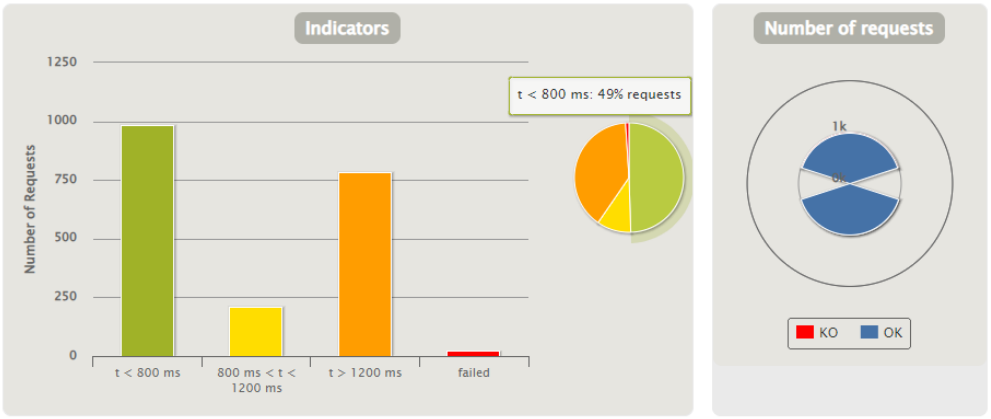
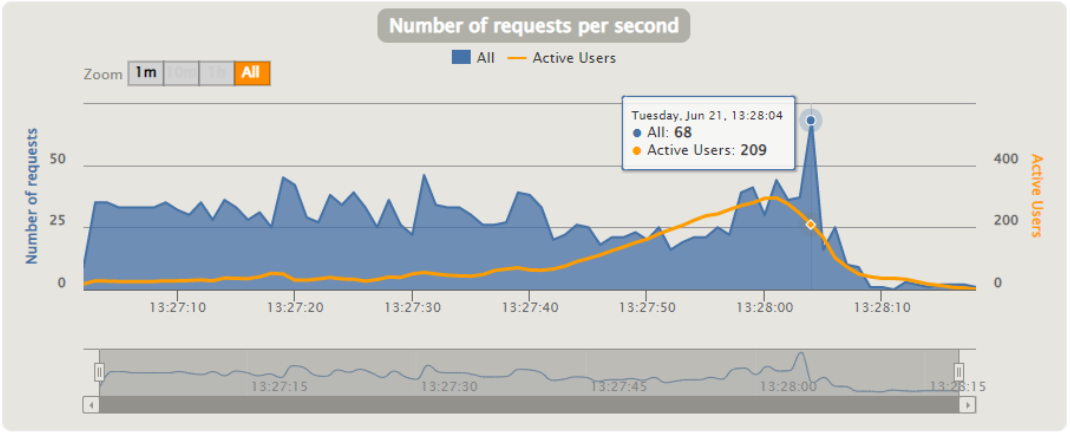


Figura 183

Número de peticiones por segundo con 1000 usuarios.



Por lo tanto, se puede notar que el servidor con usuarios menores a 1000 responde eficientemente, sin embargo, al comenzar a aumentar el número de ingresos comienza a generarse peticiones fallidas, en la tabla 29 se puede observar un resumen de los datos de las pruebas de carga.

Tabla 29

Resumen de resultados de peticiones.

Usuarios	Peticiones Totales	Máximas [Pet/s]	Éxito [%]	Fallo [%]	Respuesta t <800ms [%]	Respuesta t >1200ms [%]
100	200	15	100.0	0.0	92.0	8.0
200	400	8	100.0	0.0	100.0	0.0
500	1000	20	100.0	0.0	99.0	0.0
1000	2000	68	99.0	1.0	49.0	39.0
4000	8000	133	94.0	6.0	1.0	91.0

Pruebas de usabilidad

Para poder verificar que tan amigable es la interfaz local y remota para el usuario, se utiliza el sistema de escalas de usabilidad (System Usability Scale, SUS), este sistema cuenta con 10 preguntas estandarizada que son indicadas por Nathan, (2022):

1. ¿Creo que me gustaría usar este sistema con frecuencia?
2. ¿Encontré el sistema innecesariamente complejo?
3. ¿Pensé que el sistema era fácil de usar?
4. ¿Creo que necesitaría el apoyo de un técnico para poder utilizar este sistema?
5. ¿Descubrí que las diversas funcionalidades de este sistema estaban bien integradas?
6. ¿Pensé que había demasiadas inconsistencias en este sistema?
7. ¿Me imagino que la mayoría de la gente aprendería a usar este sistema muy rápidamente?
8. ¿Encontré el sistema muy engorroso de usar?
9. ¿Me sentí muy confiado usando el sistema?

10. ¿Necesitaba aprender muchas cosas antes de poder ponerme en marcha con este sistema?

Se evaluó a 10 usuarios que pudieron usar la aplicación de manera individual, debido a que la pagina local desplegada controla un único sistema, sin embargo, el acceso a la página pública no hubo ningún inconveniente al ingresar más de 2 usuarios a la vez, las respuestas de las preguntas realizadas se las puede observar en la tabla 30, cabe recalcar que las personas que pudieron probar el sistema, el 50% no cuenta con una preparación técnica, y el 50% restante cuentan con dicha preparación, no obstante, previo al uso se realizó una breve explicación del funcionamiento del sistema y se pudo notar que no hubo ningún inconveniente en el uso del mismo. Al finalizar las pruebas se obtiene los puntajes SUS y el promedio con un valor de 90.5/100 lo que significa que el sistema es aceptable en términos de usabilidad.

Tabla 30

Resultados de pruebas de usabilidad.

Preguntas / Usuarios	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Puntaje SUS
U1	4	1	5	1	4	1	5	1	5	1	95
U2	3	3	3	2	3	3	5	3	5	2	65
U3	4	1	5	1	5	1	5	1	5	1	97.5
U4	5	1	5	1	5	1	5	1	5	1	100
U5	4	1	4	1	5	1	4	2	5	1	90
U6	5	1	5	1	4	1	4	2	5	2	90
U7	4	1	5	1	4	1	5	2	5	1	92.5
U8	4	1	5	1	4	1	5	1	4	1	92.5
U9	5	1	4	1	5	1	4	2	5	2	90
U10	5	1	4	1	4	1	5	2	5	1	92.5
									Promedio:		90.5

Capítulo 6: Conclusiones y recomendaciones

Conclusiones.

La investigación realizada en este proyecto permitió determinar una metodología válida para la implementación de un sistema de clasificación binario. Esta consume el servicio web cognitivo ofrecido por Microsoft Azure denominado Computer Vision. A través de la FaaS de etiquetas de imágenes accede a las funcionalidades de la API para obtener etiquetas en formato JSON que son identificadas como relevantes dentro del contenido de la imagen que es capturada y suministrada al sistema, y poder así ser usadas para el proceso de identificación y clasificación.

Para cubrir la necesidad de clasificación de objetos que no son identificados o son identificados de manera incorrecta por el servicio en la nube, se implementa una red neuronal convolucional, la cual generará un modelo de aprendizaje para cada nuevo par de objetos que se desea clasificar. Para ello es necesario el uso de la técnica de transferencia de aprendizaje, reduciendo así los requerimientos de procesamiento del microordenador. La red neuronal preentrenada ResNet 50 logró cubrir esta necesidad.

El modelo de clasificación empleado por el servicio en la nube presenta valores de exactitud y de valor F de entre el 90 % al 100 %, determinando que el desempeño en su proceso de clasificación dentro del sistema es correcto para grupos en los que los objetos a clasificar pertenecen a una categoría más general. Así también, los modelos de clasificación generados por parte del servicio local presentan valores de exactitud y valor F del 95 % concluyendo que el desempeño es adecuado para cuando los objetos a clasificar pertenezcan a categorías más específicas o con características más distintivas.

Con el framework Django se implementó un sistema que permite la integración del Backend como contenedor y administrador de las funciones del servicio en la nube, así como del servicio local. Además del control de los elementos que conforman la capa a nivel físico del

sistema, también permitió la integración del Frontend el cual contiene una interfaz web local de control y supervisión y una interfaz web/móvil remota de supervisión. Ambas interfaces basadas en la biblioteca de diseño multiplataforma Bootstrap y la estructura de Dashboard presentada por gentelella.

El lenguaje de programación Python permitió la implementación de protocolos, técnicas de comunicación y transmisión de datos entre las diferentes capas de la arquitectura del sistema. Este permitió el envío de datos en formatos JSON para comunicación con la web, así como la lectura de los datos enviados desde el servicio de la nube, y el envío de información al microrordenador por medio de métodos POST para la activación de los actuadores del sistema. Por otra parte, este lenguaje de programación facilitó el uso de Web Sockets como tecnología para establecer un canal de comunicación abierto entre el cliente y el servidor, permitiendo la lectura de los sensores del sistema en tiempo real.

El uso de la plataforma como servicio PaaS Heroku permitió el despliegue de la aplicación Django en un servidor en la nube, generando un enlace público para el acceso a la interfaz web/ móvil remota para supervisión del sistema. Por su parte, el uso de la plataforma de Google FireBase y sus servicios Cloud Storage para almacenamiento de imágenes y RealTime como base de datos, dieron al sistema la capacidad de almacenamiento de información y sincronización de los datos en las interfaces de control y supervisión.

El prototipo mecánico y electrónico diseñado e implementado simula el proceso de una banda transportadora con dos actuadores como compuertas, empleadas para la clasificación de los objetos en dos grupos previamente determinados. Permitió validar de manera correcta el funcionamiento del servicio en la nube y del servicio local, ambos implementados en un microordenador Raspberry Pi 4B.

El tambor motriz y el tambor de reenvío deben poseer una alineación correcta para evitar que la banda de poliéster se incline a un solo lado y provoque un mal funcionamiento del

mecanismo de transporte de objetos. La distancia entre los extremos de los rodillos de cada tambor debe coincidir, así también deben estar correctamente lubricados sus ejes de giro para evitar travas.

Los sensores de presencia FC51 se deben calibrar mediante la variación de su resistencia de precisión para evitar la detección de objetos que se encuentran fuera del rango de detección deseada en el mecanismo. Por otro lado, la cámara Rev 1.3 no posee características para contrarrestar la saturación, por tal motivo debe ubicarse en un ambiente que controle la cantidad excesiva de luz y evitar este inconveniente para la captación y análisis incorrecto del contenido de imagen captada.

Con los resultados de las pruebas de carga, se concluye que en robustes el servidor en la nube Heroku, cumple con las necesidades del proyecto. Sin embargo, cuando el número de usuarios es mayor a 1000 las peticiones realizadas comienzan a fallar con un 1%, al contar con un error relativamente bajo se puede realizar el uso de la página adecuadamente. No obstante, al ampliar el número de usuarios a 4000 el porcentaje de fallo se eleva a 6%, y las peticiones restantes a pesar de llegar con éxito empiezan a realizarse con un tiempo de respuesta mayor. Esto genera problemas en el uso de la página, porque impide tener una correcta experiencia del servicio al usuario, cabe recalcar que debido al plan de suscripción usado en la plataforma Heroku, el número máximo de llamadas a la API en una hora es de 4500.

Una vez finalizada las pruebas de usabilidad, se obtuvo como resultado un puntaje de 90.5/100, por lo que se puede concluir que la interfaz de la página web local y remota es amigable para el usuario, el texto informativo en la página de ayuda de la interfaz local permitió también la facilidad del uso del sistema, además cabe recalcar que se pudo manejar el sistema independientemente de los conocimientos técnicos de cada uno.

Recomendaciones.

Al generar una instancia dentro del portal de Azure para el consumo del servicio web se genera un acceso por una cuenta gratuita que limita el uso de este a 20 transacciones por minuto y 5000 transacciones por mes, por ello se recomienda mantenerse dentro de ese rango de consumo o la contratación de un nuevo plan de pago para ampliar sus características.

Al generar un data set para el entrenamiento de un nuevo modelo dentro del servicio local se recomienda que este no exceda el tamaño propuesto de 50 imágenes, 25 por clase, pues provoca la ralentización o congelamiento del microordenador, pues sus características de procesamiento son limitadas.

Es recomendable que al momento de capturar la imagen para que el sistema identifique el objeto a clasificar la cámara lo enfoque en su totalidad, también se debe evitar fondos con contenido, como posters u otros elementos, además se debe contar con una iluminación adecuada para evitar problemas de identificación.

Para el uso del sistema de clasificación se debe tener en consideración las limitaciones físicas y mecánicas de la estructura, tanto en las dimensiones de los objetos, así como la velocidad a la que deben ser ingresados, procurando que el elemento anterior ya haya sido clasificado antes de un nuevo ingreso.

El microordenador Raspberry Pi debe ser alimentado con una fuente de alimentación de 12 V a 3 A para evitar su mal funcionamiento.

Trabajos futuros

La principal limitación del sistema clasificador a ser cubierta por trabajos futuros, es la de llevar a un procesador en la nube la tarea de la generación de modelos de clasificación que en este proyecto se lo realiza de manera local, pues existe límites en el data set que puede ser ingresado así como las épocas empleadas para su entrenamiento, pues esto limita al sistema al reconocimiento de objetos que poseen muchas características similares pero son de

diferentes clases, se plantea el uso de técnicas como Web Scraping para acceder a la plataforma de Google Colab y el uso de máquinas virtuales para la generación del modelo y su descarga en el microordenador, consiguiendo que sea solo necesario cargarlo y usarlo en el sistema.

La interfaz local de supervisión y control es ejecutada dentro de un servidor local generado por Django, se plantea poder llevarlo a un servidor en la nube y tener acceso al control de la máquina desde un lugar remoto, además de la generación de niveles de accesibilidad de usuarios, controlando las actividades permitidas dentro del sistema.

Finalmente se plantea poder ampliar el sistema incrementando el número de clases para cada modelo de clasificación, para lo cual será necesario la modificación de ambos servicios y de la estructura mecánica.

Acrónimos

- API Interfaz de programación de aplicaciones
- CNN Convolutian Neural Networks
- ML, Machine Learning
- PaaS Plataforma como servicio
- SaaS Software como servicio
- FaaS Función como servicio
- SLR Revisión Sistemática de la Literatura
- SMS Mapeo Sistemática de la Literatura

Bibliografía

Abad, A. C., Ligutan, D. D., Dadios, E. P., Jaeron, L., Cruz, S., Carlo, M., Rosario, D. P. Del, Nathan, J., & Kudhal, S. (2018). Fuzzy Logic-Controlled 6-DOF Robotic Arm Color-based Sorter with Machine Vision Feedback. *IJACSA) International Journal of Advanced Computer Science and Applications*, *9*(5). www.ijacsa.thesai.org

Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, *22*(5), 717–727. https://doi.org/10.1016/S0731-7085(99)00272-1

Aguila Antonio Zárate, Aguila Reyes Alejandro, Servando Ariel, Aguila Zárate, & Mendoza Blanco Fernando. (2009). *Análisis de lA visión humana orientado al diseño arquitectónico analysis of human vision orientated toward architectur Al design* (Issue 1). http://erevistas.saber.ula.ve/index.php/ecodiseno/article/view/3888/0

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2018). Understanding of a convolutional neural network. *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, *2018-January*, 1–6. https://doi.org/10.1109/ICENGTECHNOL.2017.8308186

Amaya Zapata, S., Pulgarín Velásquez, D., & Torres Pardo, Í. D. (2016). Desarrollo e Implementación de un Sistema de Visión Artificial Basado en Lenguajes de Uso Libre para un Sistema Seleccionador de Productos de un Centro Integrado de Manufactura (CIM). *Lámpsakos*, *15*, 43. https://doi.org/10.21501/21454086.1702

Barrios, J. (2022). *Redes neuronales convolucionales* . https://www.juanbarrios.com/redes-neurales-convolucionales/

Basurto, B. (2013). *Diseño de bandas transportadoras tubulares* [ESPOL]. https://www.dspace.espol.edu.ec/handle/123456789/25153

Bernacki, J. (2021). Robustness of digital camera identification with convolutional neural networks. *Multimedia Tools and Applications*, *80*(19), 29657–29673. https://doi.org/10.1007/S11042-021-11129-Y/TABLES/10

Bharatia, D., Ambawane, P., & Rane, P. (2019). Smart Electronic Stick for Visually Impaired using Android Application and Google's Cloud Vision. *2019 Global Conference for Advancement in Technology, GCAT 2019*. https://doi.org/10.1109/GCAT47503.2019.8978303

Bocci, A., Forti, S., Ferrari, G. L., & Brogi, A. (2021). Secure FaaS orchestration in the fog: how far are we? *Computing*, *103*(5), 1025–1056. https://doi.org/10.1007/S00607-021-00924-Y/TABLES/3

Brown, N., Cambruzzi, J., Cox, P. J., Davies, M., Dunbar, J., Plumbley, D., Sellwood, M. A., Sim, A., Williams-Jones, B. I., Zwierzyna, M., & Sheppard, D. W. (2018). Big Data in Drug Discovery. *Progress in Medicinal Chemistry*, *57*(1), 277–356. https://doi.org/10.1016/BS.PMCH.2017.12.003

Coccoli, M., De Francesco, V., Fusco, A., & Maresca, P. (2022). A cloud-based cognitive computing solution with interoperable applications to counteract illegal dumping in smart cities. *Multimedia Tools and Applications*, *81*(1), 95–113. https://doi.org/10.1007/S11042-021-11238-8/TABLES/3

De La Cruz, A. E., & Donoso, J. F. (2016). *Diseño y construcción de una máquina didáctica clasificadora de objetos mediante visión artificial para el Laboratorio de Automatización Industrial de Procesos Mecánicos de la Facultad de Ingeniería Mecánica* [EPN]. https://bibdigital.epn.edu.ec/handle/15000/16486

Django. (2022). *The web framework for perfectionists with deadlines*.

https://www.djangoproject.com/

Faisal, M., Albogamy, F., Elgibreen, H., Algabri, M., & Alqershi, F. A. (2020). Deep Learning and Computer Vision for Estimating Date Fruits Type, Maturity Level, and Weight. *IEEE Access*, *8*, 206770–206782. https://doi.org/10.1109/ACCESS.2020.3037948

Forbo. (2022). *Recomendaciones para la construcción de instalaciones*. https://forbo.blob.core.windows.net/forbodocuments/7378/305_fms_recomendaciones_para_la_construccion_de_instalaciones_es.pdf

Garrett, J. J. (2005). *Ajax: A New Approach to Web Applications*. http://www.adaptivepath.com/publications/essays/archives/000385print.php

GEICOM. (2020). *Tipos de sorters o clasificadores que deberías conocer*. https://blog.gieicom.com/tipos-de-sorters-o-clasificadores-que-deberias-conocer

Ghosh, A., Sufian, A., Sultana, F., Chakrabarti, A., & De, D. (2020). Fundamental Concepts of Convolutional Neural Network. *Intelligent Systems Reference Library*, *172*, 519–567. https://doi.org/10.1007/978-3-030-32644-9_36

Gnana, A., Kumar, S., Aathisha, S., Dharani, S., & Revathi, N. (2019). *Machine Vision Technique Based Smart Fruit Sorter*. https://doi.org/10.1109/JSEN.2016.2580221

Holguín, C. M., Osorio, J. A. C., & Osorio, J. A. C. (2014). Automatic recognition system of fruits based computer vision. *Ingeniare*, *22*(4), 504–516. https://doi.org/10.4067/S0718-33052014000400006

Hope, R. (2017). *Understanding the differences between AI, machine learning, and deep learning*. https://deeplearning.lipingyang.org/wp-content/uploads/2016/11/Understanding-the-differences-between-AI-machine-learning-and-deep-learning-TechRepublic.pdf

Huang, J., Shang, Y., & Chen, H. (2019). Improved Viola-Jones face detection algorithm based on HoloLens. *Eurasip Journal on Image and Video Processing*, *2019*(1), 1–11. https://doi.org/10.1186/S13640-019-0435-6/TABLES/1

IBM. (2022). *What is Computer Vision?* | *IBM*. https://www.ibm.com/topics/computer-vision

ImageNet. (2021, March 11). *ImageNet*. https://www.image-net.org/

Keras. (2022). *Transfer learning & fine-tuning*. https://keras.io/guides/transfer_learning/

Kolivand, H., Joudaki, S., Sunar, M. S., & Tully, D. (2021). A new framework for sign language alphabet hand posture recognition using geometrical features through artificial neural network (part 1). *Neural Computing and Applications*, *33*(10), 4945–4963. https://doi.org/10.1007/S00521-020-05279-7/FIGURES/14

Li, X., Dai, B., Sun, H., & Li, W. (2019). Corn Classification System based on Computer Vision. *Symmetry 2019, Vol. 11, Page 591*, *11*(4), 591. https://doi.org/10.3390/SYM11040591

Lie, W., Jiang, B., & Zhao, W. (2020). Obstetric imaging diagnostic platform based on cloud computing technology under the background of smart medical big data and deep learning. *IEEE Access*, *8*, 78265–78278. https://doi.org/10.1109/ACCESS.2020.2988563

Lim, S. H. (2021). Understanding Recurrent Neural Networks Using Nonequilibrium Response Theory. *Journal of Machine Learning Research*, *22*, 1–48. http://jmlr.org/papers/v22/20-620.html.

Marini, F. (2009). Neural Networks. *Comprehensive Chemometrics*, *3*, 477–505. https://doi.org/10.1016/B978-044452701-1.00128-9

Melnikov, A., & Fette, I. (2011, December). *The WebSocket Protocol*. https://datatracker.ietf.org/doc/html/rfc6455

Microsoft. (2022a). *Computer Vision | Microsoft Azure*. https://azure.microsoft.com/en-in/services/cognitive-services/computer-vision/#features

Microsoft. (2022b). *Computer Vision documentation - Quickstarts, Tutorials, API Reference - Azure Cognitive Services.* https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/

Moncada, G. (2018). *SISTEMA ROBÓTICO CLASIFICADOR DE OBJETOS CON VISIÓN ARTIFICIAL* [Universidad Técnica Federico Santa María]. https://repositorio.usm.cl/bitstream/handle/11673/42363/3560901544160UTFSM.pdf?sequence=

Nathan, T. (2022). *How To Use The System Usability Scale (SUS) To Evaluate The Usability Of Your Website - Usability Geek*. https://usabilitygeek.com/how-to-use-the-system-usability-scale-sus-to-evaluate-the-usability-of-your-website/

OpenCV. (2022). *About - OpenCV*. https://opencv.org/about/

Paulus, I., De Busscher, R., & Schrevens, E. (1997). *Use of Image Analysis to Investigate Human Quality Classification of Apples - [PDF Document]*. https://vdocuments.mx/use-of-image-analysis-to-investigate-human-quality-classification-of-apples.html

Peršak, T., Viltužnik, B., Hernavs, J., & Klančnik, S. (n.d.). *Vision-Based Sorting Systems for Transparent Plastic Granulate*. https://doi.org/10.3390/app10124269

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, *64*, 1–18. https://doi.org/10.1016/J.INFSOF.2015.03.007

Python. (2022). *What is Python?* https://www.python.org/doc/essays/blurb/

Quilloy, E. P., Suministrado, D. C., & Bato, P. M. (2018). *Single-line automated sorter using mechatronics and machine vision system for Philippine table eggs*. *13*(17), 918–926. https://doi.org/10.5897/AJAR2018.13113

RaspberryPi. (2022). *Raspberry Pi 4 Computer Model B*. www.raspberrypi.org

Salmerón Rubio, J. (2020). *Desarrollo de aplicación web basada en FaaS con .NET Core Evolution desde aplicación monolítica*. UNIVERSIDAD POLITECNICA DE MADRID.

Santra, A., & Christy, C. (2012). Genetic Algorithm and Confusion Matrix for Document Clustering . *IJCSI International Journal of Computer Science Issues, 9*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.2710&rep=rep1&type=pdf

Song, Y., Li, Q., Feng, D., Zou, J. J., & Cai, W. (2016). Texture image classification with discriminative neural networks. *Computational Visual Media 2016 2:4*, *2*(4), 367–377. https://doi.org/10.1007/S41095-016-0060-6

Subbulakshmi, S., Saji, A. E., & Chandran, G. (2020). Methodologies for selection of Quality Web Services to Develop Efficient Web Service Composition. *Proceedings of the 4th International Conference on Computing Methodologies and Communication, ICCMC 2020*, 238–244. https://doi.org/10.1109/ICCMC48092.2020.ICCMC-00045

Tang, Y. X., Tang, Y. B., Peng, Y., Yan, K., Bagheri, M., Redd, B. A., Brandon, C. J., Lu, Z., Han, M., Xiao, J., & Summers, R. M. (2020). Automated abnormality classification of chest radiographs using deep convolutional neural networks. *Npj Digital Medicine 2020 3:1*, *3*(1), 1–8. https://doi.org/10.1038/s41746-020-0273-z

Ting, S.-H., Ng, Y.-J., & Neelam, M. (2021). *Neelam MahaLakshmi (2021) Aspects of Artificial Intelligence In*. *1*. https://www.researchgate.net/publication/358119068

Valipoor, M. M., & de Antonio, A. (2022). Recent trends in computer vision-driven scene understanding for VI/blind users: a systematic mapping. *Universal Access in the Information Society*, *1*, 1–23. https://doi.org/10.1007/S10209-022-00868-W/FIGURES/7

Vermeire, T., Brughmans, D., Goethals, S., de Oliveira, R. M. B., & Martens, D. (2022). Explainable image classification with evidence counterfactual. *Pattern Analysis and Applications*, *25*(2), 315–335. https://doi.org/10.1007/S10044-021-01055-Y/TABLES/5

Wang, J. (2020). OCT Image Recognition of Cardiovascular Vulnerable Plaque Based on CNN. *IEEE Access*, *8*, 140767–140776. https://doi.org/10.1109/ACCESS.2020.3007599

Weiss, K., Khoshgoftaar, T. M., & Wang, D. D. (2016). A survey of transfer learning. *Journal of Big Data*, *3*(1), 1–40. https://doi.org/10.1186/S40537-016-0043-6/TABLES/6

Wierzbiński, M., Pławiak, P., Hammad, M., & Acharya, U. R. (2021). Development of accurate classification of heavenly bodies using novel machine learning techniques. *Soft Computing*, *25*(10), 7213–7228. https://doi.org/10.1007/S00500-021-05687-4/TABLES/8

Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., & Wang, C. (2018). Machine Learning and Deep Learning Methods for Cybersecurity. *IEEE Access*, *6*, 35365–35381. https://doi.org/10.1109/ACCESS.2018.2836950

Zhao, J., Hao, S., Dai, C., Zhang, H., Zhao, L., Ji, Z., & Ganchev, I. (2022). Improved Vision-Based Vehicle Detection and Classification by Optimized YOLOv4. *IEEE Access*, *10*, 8590–8603. https://doi.org/10.1109/ACCESS.2022.3143365

Zhihong, C., Hebin, Z., Yanbo, W., Binyan, L., & Yu, L. (2017). A vision-based robotic grasping system using deep learning for garbage sorting. *Chinese Control Conference, CCC*, 11223–11226. https://doi.org/10.23919/CHICC.2017.8029147

Apéndices