UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

Proyecto de investigación previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

TRABAJO DE TITULACIÓN

"IMPLEMENTACIÓN DE UN SISTEMA PARA EL RECONOCIMIENTO Y CLASIFICACIÓN DE LA CONTAMINACIÓN SUPERFICIAL DE LA LAGUNA DE COLTA MEDIANTE ALGORITMOS DE INTELIGENCIA ARTIFICIAL"

Autor:

William Javier Yugsi Apupalo

Tutor:

Mgs. José Luis Jinez Tapia

Riobamba – **Ecuador**

Año 2020

Los miembros del tribunal de graduación del proyecto de investigación de título: **"IMPLEMENTACIÓN DE UN SISTEMA PARA EL RECONOCIMIENTO Y CLASIFICACIÓN DE LA CONTAMINACIÓN SUPERFICIAL DE LA LAGUNA DE COLTA MEDIANTE ALGORITMOS DE INTELIGENCIA ARTIFICIAL."**, presentado por: William Javier Yugsi Apupalo, dirigido por: Mgs. José Luis Jinez Tapia.

Una vez escuchada la defensa oral y revisado el informe final del proyecto de investigación con fines de graduación escrito en la cual se ha constatado el cumplimiento de las observaciones realizadas, remite la presente para uso y custodia en la biblioteca de la Facultad de Ingeniería de la UNACH.

Para constancia de lo expuesto firman.

Mgs. Deysi Inca ……

Presidente del Tribunal **Firma**

Mgs. Giovanny Cuzco ……

Miembro del Tribunal **Firma**

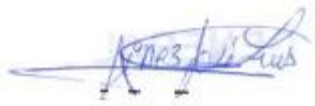
PhD. Leonardo Rentería ……

Miembro del Tribunal **Firma**

DECLARACIÓN EXPRESA DE TUTORÍA

En calidad de tutor del tema de investigación **"IMPLEMENTACIÓN DE UN SISTEMA PARA EL RECONOCIMIENTO Y CLASIFICACIÓN DE LA CONTAMINACIÓN SUPERFICIAL DE LA LAGUNA DE COLTA MEDIANTE ALGORITMOS DE INTELIGENCIA ARTIFICIAL."** realizado por el Sr. William Javier Yugsi Apupalo, para optar por el título de Ingeniero en Electrónica y Telecomunicaciones, considero que reúne los requisitos y méritos suficientes para ser sustentada públicamente y evaluada por el jurado examinador que se designe.

Riobamba, diciembre 2020



Mgs. José Luis Jinez Tapia ……

C.I. 0602899007 **Firma**

TUTOR:

AUTORÍA DE LA INVESTIGACIÓN

La responsabilidad del contenido de este proyecto de graduación corresponde exclusivamente a: **William Javier Yugsi Apupalo, Mg. Marlon Jose Luis Jinez Tapia,** y el patrimonio intelectual de la misma a la Universidad Nacional de Chimborazo.

…….

William Javier Yugsi Apupalo

C.I. 180479379-0

DEDICATORIA

Este trabajo de fin de grado va dedicado en primer lugar a Dios, por haberme dado fuerza en momentos difíciles y no decaer en presencia de adversidades, a mi madre *Narciza de Jesús* quien fue el principal apoyo para inicio de mi vida profesional, quien me acompañado en todo momento y con sus consejos ha sabido forjarme como persona, quien con su gran corazón me lleva a admirarla cada día más.

A mi padre *Marcelo*, a mi esposa *Ana Gabriela*, a mi hijo *Dylan*, a mi abuelita mamá *María* y a toda mi familia que son personas que me ha brindado amor y quienes con sus palabras de aliento me impulsaron a ser perseverante y cumplir con mis objetivos anhelados.

William Yugsi A.

AGRADECIMIENTOS

Agradezco en primer lugar Dios por sus bendiciones, a mi madre quien con su apoyo incondicional hizo que surgiera en los estudios y pueda culminar una etapa más de mi vida, que con su carisma y su buen corazón sabría entenderme y tener la sabiduría para guiarme por el camino del bien, a mi padre *Marcelo Cando*, aunque no llevemos la misma sangre para mí siempre será el papá que nunca tuve, por el apoyo y confianza que siempre sostuvo en mí.

A mi esposa *Ana Gabriela* e hijo *Dylan* a quienes amo con todo mi corazón y son motivo de mi superación, a mis suegros papá *Roberto* y mamá *Magdalena* por el apoyo, por saber educar y cuidar de mi hijo todo el tiempo que estuve ausente.

A mis docentes, a mi tutor *José Jinez, Geovanny Cuzco y Bryan Guevara* que supieron aportar sus conocimientos y ayuda para concluir con éxito el proyecto.

A mis amigos y compañeros que supieron compartir conocimientos y pasar momentos inolvidables.

William Yugsi A.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La Laguna de Colta siendo un atractivo natural y uno de los lugares con gran afluencia de turistas, toma una importante característica en cuanto se refiere al aspecto visual. Sin embargo, la Laguna se puede ver comprometida por contaminación superficial (residuos sólidos contaminantes), los cuales además de afectar no solo a la Laguna en mención, sino también, a todo el lugar cercano a las orillas de la Laguna, es por ello que se ha propuesto realizar el presente proyecto de investigación.

En el presente proyecto se expone el desarrollo de un algoritmo del campo de la inteligencia artificial de redes neuronales en donde se hace aplicación del *Deep Learning* para el reconocimiento y clasificación de objetos contaminantes presentes en la superficie de la Laguna de Colta, a partir de imágenes de video obtenidas a través de una cámara FPV que es movilizada por un vehículo aéreo no tripulado (UAV).

En concreto el sistema propone seguir un método robusto para detectar objetos contaminantes (botellas plásticas, envases metálicos y totora) basado en CNN (Convolutional Neural Network) y recopilando una base de datos de imágenes específicas y con técnicas de *Transfer Learning*, se propone que alcance un rendimiento casi en tiempo real y una precisión satisfactoria. Para superar los desafíos de construir un modelo preciso de detección de objetos contaminantes, se transformó la arquitectura principal de la red AlexNet pre-entrenada a una red neuronal convolucional basada en una región más rápida (Faster R-CNN) en el entorno de MATLAB.

En consecuencia, el sistema permitirá la gestión automática de la concentración de contaminación superficial y proporcionar resultados aceptables en términos de precisión y tiempo de ejecución. Por último, se ha realizado dos Interfaz Gráficas (Matlab, Unity) donde se visualizó los resultados y la posición de los objetos detectados arrojados por el sistema en tiempo real.

Palabras clave: Aprendizaje Automático, Aprendizaje Profundo, Algoritmos, Redes Neuronales Convolucionales, Faster R-CNN.

ABSTRACT

The Laguna de Colta, being a natural attraction and one of the places with a large influx of tourists, takes on an important characteristic as regards the visual aspect. However, the Lagoon can be compromised by surface contamination (polluting solid waste), which in addition to affecting not only the Lagoon in question, but also the entire place near the shores of the Lagoon, for this reason it has been proposed to carry out this research project.

In this project the development of an algorithm in the field of artificial intelligence of neural networks is exposed, where Deep Learning is applied for the recognition and classification of polluting objects present on the surface of the Laguna de Colta, from images video obtained through an FPV camera that is mobilized by an unmanned aerial vehicle (UAV).

Specifically, the system proposes to follow a robust method to detect contaminating objects (plastic bottles, metal containers and reeds) based on CNN (Convolutional Neural Network) and compiling a database of specific images and with Transfer Learning techniques, it will work in near real-time performance and satisfactory accuracy. To overcome the challenges of building an accurate contaminating object detection model, the core architecture of the pre-trained AlexNet network was transformed to a faster region-based convolutional neural network (Faster R-CNN) in the MATLAB environment.

Consequently, the system will allow the automatic management of the concentration of surface contamination and provide acceptable results in terms of precision and execution time. Finally, two Graphical Interfaces (Matlab, Unity) have been created where the results and the position of the detected objects thrown by the system were visualized in real time.

Keywords: Machine Learning, Deep Learning, Algorithms, Convolutional Neural Networks, Faster R-CNN.

Reviewed by:
Danilo Yépez Oviedo
English professor UNACH

INTRODUCCIÓN

La población del Ecuador según el Instituto Nacional de Estadística y Censos es de 17.254.840 millones de habitantes (Riofrío, 2017), registrándose que un 77% de los hogares elimina la basura a través de carros recolectores y el restante 23% la elimina de diversas formas, así por ejemplo la arroja a terrenos baldíos o quebradas, la quema, la entierra, la deposita en ríos, lagos, lagunas, acequias o canales, etc. (PNGIDS, 2017).

La acumulación de residuos o desechos por parte de las personas en ambientes acuáticos es el resultado de la naturaleza y/o actividades antrópicas. La creciente y cada vez más compleja actividad humana (industria y doméstica) durante las últimas décadas ha impuesto un serio riesgo al ambiente y consecuentemente a la salud humana.

Muchas veces, cuando hablamos de la tecnología "Inteligencia Artificial" (IA), lo primero que se nos viene a la mente es la imagen de un robot o de tecnologías muy sofisticadas, para ser más exactos, es la simulación de procesos de inteligencia humana por parte de máquinas, especialmente sistemas informáticos. Hoy en día se habla acerca de los términos "Machine Learning" (MathWorks, 2016) y "Deep Learning" (Varma & Das, 2017) (Aprendizaje Autónomo, Aprendizaje Profundo), que es una subdisciplina de la IA, producto de las ciencias de la computación y las neurociencias.

(Sanz Sardina, 2014), Parte de un sistema de reconocimiento y clasificación de objetos el cual tiene como fin tratar de emular la forma como los humanos percibimos la información visual mediante el uso de una videocámara, los diferentes métodos de visión existentes para el reconocimiento de objetos buscan patrones cualitativos para el reconocimiento de los objetos de interés.

El presente trabajo de investigación tiene como objetivo determinar el nivel de contaminación superficial de la laguna de Colta mediante la implementación de un sistema, orientado a la detección y clasificación de objetos y/o desechos sólidos de manera autónoma, el procesamiento se lo elabora a partir de una secuencia de imágenes de escenas de la laguna provenientes de una videocámara, el sistema automáticamente tiene la labor de detectar e identificar los contaminantes superficiales (botellas plásticas, envases metálicos y totora) y arrojar los niveles de contaminación según el tipo de contaminante.

La implementación de este sistema está sujeta a ambientes no estructurados en donde los objetos a detectar poseen matrices con baja resolución. El presente documento de investigación se divide en capítulos, los cuales están estructurados de la siguiente manera:

- **Capítulo 1: Introducción**

  Se realizará una breve explicación de la descripción de este proyecto, donde se puede encontrar el planteamiento del problema y los objetivos a alcanzar, así también como la estructura del contenido.

- **Capítulo 2: Estado del Arte**

  Explicación teórica acerca del estado actual de la presente investigación y sobre el Deep Learning (Aprendizaje Profundo). Así entrando en detalle en diferentes temas como los tipos de redes neuronales que se irán mencionando a lo largo del proyecto, el *Transfer Learning* y el *Data Augmentation*.

- **Capítulo 3: Metodología**

  Explicación detallada de todo el procedimiento realizado para el desarrollo del sistema para el reconocimiento y clasificación de la contaminación superficial, así como los errores y problemas detectados, hasta llegar a la solución final.

- **Capítulo 4: Conclusiones y Recomendaciones**

  Por último, se tratarán las conclusiones adquiridas, donde se evaluará la eficiencia del método propuesto a la hora de detectar y clasificar objetos contaminantes.

El punto de partida fue la fundamentación teórica que ha sido la base de la investigación permitiendo profundizar el conocimiento en temas de sistemas de medición y la implementación desde otra perspectiva, utilizando los antecedentes de otras investigaciones que sirvan de base para profundizar la temática utilizando de una serie de artículos científicos relacionados al tema de investigación.

CAPÍTULO I

1.1. PLANTEAMIENTO DEL PROBLEMA

¿Cómo implementar un sistema para el reconocimiento y clasificación de la contaminación superficial de la Laguna de Colta mediante algoritmos de inteligencia Artificial?

Laguna de Colta siendo como uno de los mejores atractivos de la provincia de Chimborazo y tiene como objetivo atraer a turistas tanto nacionales y extranjeros para que sean partícipes de sus costumbres, tradiciones y actividades de su localidad. Al mismo tiempo presenta varios factores negativos como es la deficiente gestión en planes de manejo de residuos sólidos, limitado control residual, el deficiente conocimiento sobre sistemas de medición de residuos contaminantes, la débil organización e inexistencia de proyectos de inversión, todas estas falencias han provocado que la laguna de Colta muestre un aspecto no amigable con el medio ambiente y se desarrolle turísticamente.

En la actualidad la tecnología se ha convertido en un instrumento primordial para la solución de problemas es por ello por lo que mediante su utilización podemos llegar a cualquier lugar y obtener información.

En el Ecuador las recreaciones turísticas no cuentan con antecedentes sobre el uso de sistemas que sirvan para el reconocimiento y clasificación de residuos contaminantes de Lagos y Lagunas, por lo que utilizar aeromodelos es una solución novedosa y viable en poder informar de la cantidad de desechos contaminantes superficiales y el material que muestra mayor concentración de contaminación a las autoridades ambientales interesadas.

El propósito general de este proyecto consiste en la implementación de un sistema de medición utilizando un vehículo aéreo no tripulado (UAV) (Corrigan, 2019) para medir los niveles de contaminación superficial en la laguna de Colta, el cual utiliza una videocámara que es desplazada en un UAV y de esta manera tomar muestras de imágenes, dichos datos sean intervenidos y procesados en la CPU, así poder arrojar y visualizar información contundente en tiempo real de los agentes contaminantes existentes.

Además de realizar mediciones estadísticas del factor contaminante, se evalúa su comportamiento en el espacio y el tiempo, asociándolo con los fenómenos meteorológicos, composición química y origen, los cuales permitan orientar estrategias de control y realizar seguimiento por parte de las autoridades ambientales interesadas.

Se establece que a su vez dicha información haya sido analizada e interpretada por autoridades ambientales interesadas las mismas que podrán gestionar una propuesta que permita concientizar a las personas locales y visitantes de la importancia que es conservar la laguna sin contaminación.

1.2. OBJETIVOS

1.2.1. OBJETIVO GENERAL

- Implementar un sistema para el reconocimiento y clasificación de la contaminación superficial de la laguna de Colta mediante algoritmos de inteligencia artificial.

1.2.2. OBJETIVOS ESPECÍFICOS

- Desarrollar un algoritmo de inteligencia artificial capaz de clasificar materiales contaminantes como plástico, metal y totora para su interpretación porcentual.
- Implementar una plataforma de comunicación que permita adquirir en tiempo real datos como imágenes, sensores bio-ambientales, parámetros del UAV para procesos en el ordenador.
- Desarrollar un interfaz de usuario para visualizar la información geo referencial de los datos contractados.

CAPÍTULO II

2.1.ESTADO DEL ARTE

Para llevar a cabo la realización de la investigación es necesario revisar los estudios previos que se han elaborado y tengan una relación con el presente tema, de esta manera obtener una fundamentación clara, precisa y acertada del estado actual de la investigación.

En esta sección se dará una visión global sobre el estado actual del sistema para el reconocimiento y clasificación de objetos en imágenes, mediante el uso de algoritmos de inteligencia artificial, así como los procesos, pasos previos y algoritmos más utilizados para esta labor.

Los autores (Rendón, Josué, Jonathan, Jorge, & Andrea, 2018), en el presente artículo denominado “Aprendizaje Profundo para la Identificación de Objetos en Robótica Móvil”. Tienen como objetivo realizar un estudio comparativo con dos tipos de redes neuronales convolucionales como son AlexNet y VGGNE aplicadas al aprendizaje profundo.

1. AlexNet fue creada para un concurso llamado ImageNet Large Scale Visual Recognition Competition (ILSVRC), que es una competencia donde participan para encontrar el mejor modelo de visón por computadora para tareas como detección, localización y clasificación de imágenes, las cuales está formada por cinco capas convolucionales, capas de agrupación máxima, capas eliminadas, y tres capas totalmente conectadas en la red neuronal de tipo convolución.

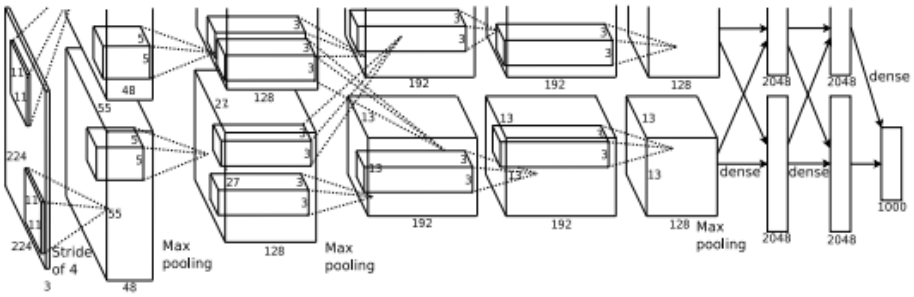


Figura 1. Arquitectura de la Red AlexNet

Fuete: (Ketkar, 2017)

2. Red neuronal convolucional VGGNET, está formada por 16 capaz y es utilizada por el grupo de Geometría Visual de la Universidad de Oxford para un concurso llamado

ImageNet Large Scale Visual Recognition Competition (ILSVRC). Esta red contiene una base de entrenamiento de 254*254 imágenes en RGB a través de 5 bloques de capas convolucionales donde cada bloque está compuesto por un número creciente de filtros de 3x3.

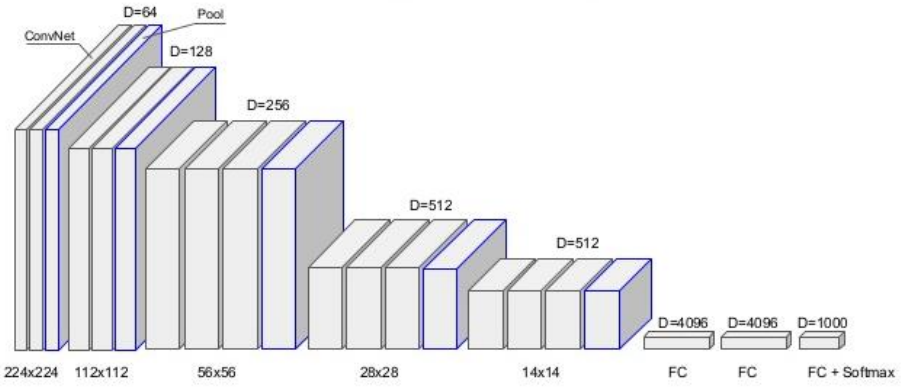


Figura 2. Topología clásica de CNN-VGGNET

Fuente: (Das V. &., 1017)

La investigación mencionada se suma a la creciente evidencia en función al desempeño en alcanzar la precisión más convincente para la detección de imágenes al utilizar una red neuronal convolucional, para AlexNet el procesamiento y la identificación del objeto fue más rápido que el de VGG16, a través de un conjunto de pruebas independientes coincidió con el nivel de rendimiento en clasificación. Esto se debió a la cantidad de capas que contienen cada una de ellas, de tal manera afecta el tiempo para la obtención de una respuesta.

Por su parte Eigen, Zhang, Mathieu, Fergus, & LeCun (2014), en el artículo: "Reconocimiento, localización y detección integrados por redes convolucionales", emplea un nuevo enfoque de aprendizaje profundo (Deep Learning) y tiene como propósito el mostrar que la capacitación de una red convolucional para clasificar, localizar y detectar objetos puede mejorar la precisión de la clasificación, la detección y la localización de todas las tareas en una sola ConvNet (Red Convolucional). A diferencia de otros enfoques de ventana deslizante, ConvNet es inherentemente eficiente cuando se aplica de forma deslizante porque naturalmente comparten los cálculos comunes a las regiones, esto amplía la salida de cada capa para cubrir el nuevo tamaño de imagen, produciendo finalmente un

mapa de predicciones de clase de salida, con una ubicación espacial para cada "ventana" (campo de visión) de entrada.

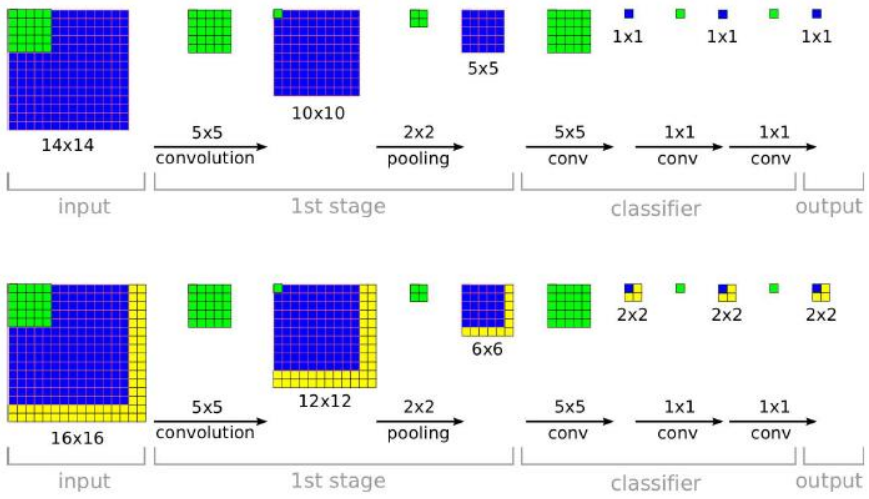


Figura 3. La eficiencia de ConvNets para la detección durante el entrenamiento

Autor: (Eigen et al., 2014)

La ConvNet presenta un enfoque de acumulación de cuadros delimitadores previstos que se puede aplicar de abajo hacia arriba, de modo que los cálculos comunes a las ventanas vecinas solo deben realizarse una vez, de manera que las ultimas capas de la arquitectura son capas lineales totalmente conectadas por ende las capas son reemplazadas efectivamente por operaciones de convolución con núcleos de extensión espacial 1x1. La ConvNet completa es, entonces, simplemente una secuencia de operaciones, agrupación máxima y operaciones de umbrales.

De acuerdo al estudio de los siguientes autores (B, Liu, Tuzel, & Xiao, 2017) en el artículo: "R-CNN para la detección de objetos pequeños" propone evaluar el algoritmo de detección de objetos que sea capaz de mejorar el rendimiento de la detección de objetos pequeños en imágenes, a través de una ampliación del algoritmo R-CNN, que está conformada con una red de propuestas de regiones y un modelado de contexto cuidadosamente diseñados.

La red neuronal consta de tres subredes donde la primera toma la región de la propuesta como entrada, la segunda toma la región del contexto como entrada y la última toma la concatenación de las salidas de las otras como entrada y calcula la puntuación final de

clasificación. Llamamos a esta red neuronal ContextNet y la estructura se muestra en la Figura 4.

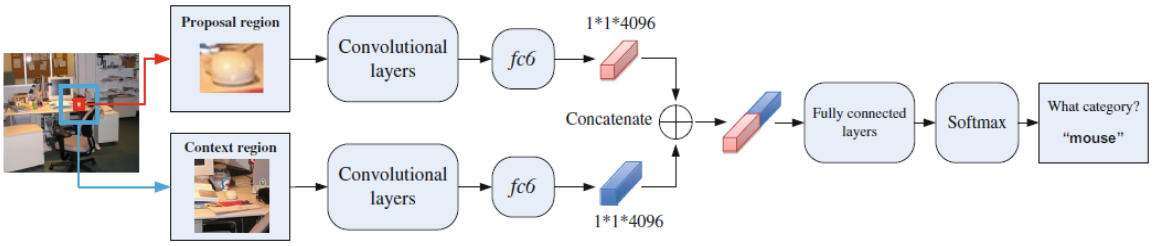


Figura 4. ContexNet: la de neuronal para integrar información de contexto

Fuente: (B et al., 2017)

Este trabajo realiza las siguientes aportaciones:

- ✓ Propone un conjunto de datos que contiene diversos objetos pequeños para facilitar el estudio de la aplicabilidad de detectores de objetos basados en el aprendizaje profundo del estado de la técnica para la detección de pequeños objetos en la imagen.
- ✓ El diseño de experimento sistemático y comparación de rendimiento, aumentando el algoritmo R-CNN, lo que aumenta el rendimiento de la detección de objetos pequeños en un 29,8% en el conjunto de datos de referencia.

El algoritmo de detección de objetos basado en el aprendizaje profundo logra una mejora de rendimiento similar con respecto al enfoque convencional para la detección de objetos pequeños como para la detección de objetos grandes.

Por último, los autores (Massé, Lathuilì Ere, Mesejo, & Horaud, 2019), en el paper titulado "Detección de objetos en videos más allá del campo de visión de la cámara" tiene por objetivo identificar los problemas de detección de objetos de interés en un video y en la estimación de sus ubicaciones, únicamente a partir de las indicaciones de las personas presentes en el video, es decir que los objetos se pueden ubicar indistintamente dentro o fuera del campo de visión de la cámara.

A continuación, se presentan las arquitecturas basadas en convolución codificador / decodificador.

a) **La heurística sin aprender:** La detección de máximos locales se realiza directamente en una combinación de calor, mapas, mirada y escritura sobre las regiones que se activan (cerca de uno) en múltiples mapas de calor de la mirada son constantemente delante de la cabeza de alguien y tienen una alta probabilidad de contener un objeto.
b) **Bases de referencia basado en el aprendizaje:** Definimos algunos modelos de regresión simple. Aprenden una regresión que significa mapas de calor de la mirada.
c) **Arquitecturas codificador / decodificador:** Se han utilizado para muchas tareas de visión por ordenador, donde el objetivo es llevar a cabo una regresión entre los espacios de alta dimensión.

El paper mencionado indica la representación espacial novedosa de las direcciones de la mirada adoptando una perspectiva de vista superior. Además, que se desarrollan varias redes de codificador / decodificadores convolucionales para predecir ubicaciones de objetos y compararlas con heurísticas y con enfoques clásicos basados en el aprendizaje.

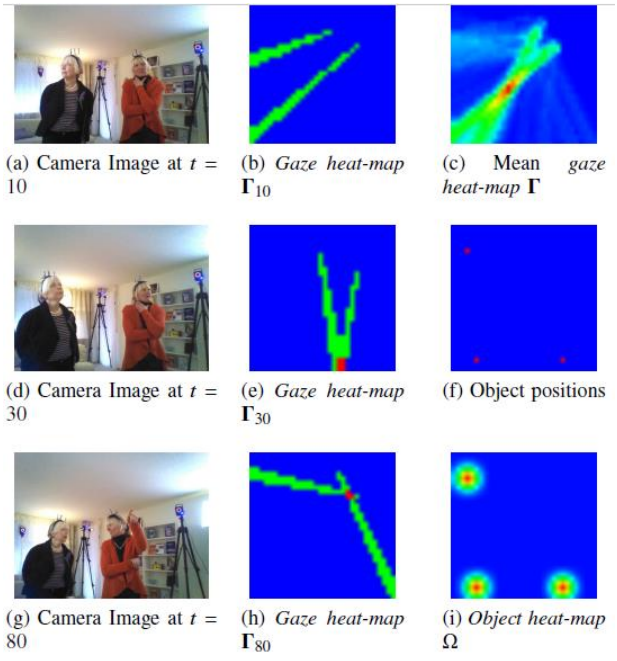


Figura 5. Representaciones del mapa de calor utilizando una secuencia extraída del conjunto de datos Vernissage

Fuente: (Massé et al., 2019)

2.2.FUNDAMENTACIÓN TEÓRICA

En el siguiente apartado se detalla la teoría necesaria para el desarrollo del proyecto.

2.2.1. Visión Artificial

La visión artificial o también llamada visión por computadora es un campo que incluye métodos para adquirir, procesar, analizar y comprender imágenes del mundo real para producir información numérica o simbólica, por ejemplo, en forma de decisiones (Franklin, 2016).

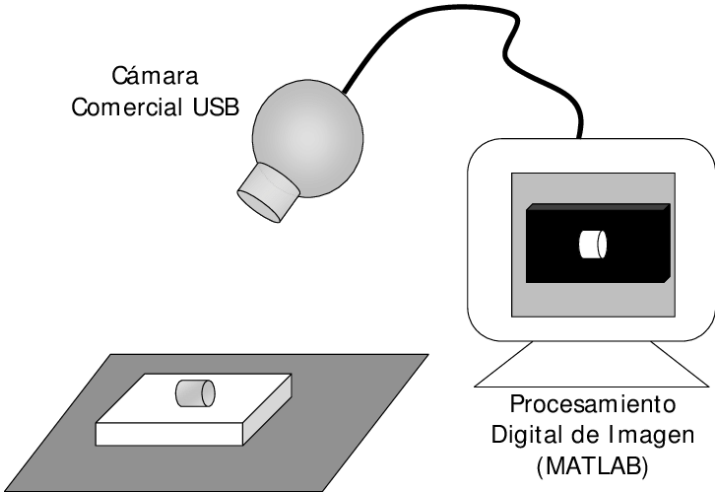


Figura 6. Sistema de visión artificial

Fuente: (Reyes, 2015)

Este campo interdisciplinario simula y automatiza estos sistemas de visión humana utilizando sensores, computadoras y algoritmos de aprendizaje autónomo, que a su vez es la teoría que subyace en la capacidad de los sistemas de inteligencia artificial para ver y comprender su entorno (Computational Learning Theory, 2019). Un sistema de visión artificial está compuesto por las siguientes etapas:

- Adquisición de Imágenes
- Pre procesamiento

- Segmentación
- Representación y descripción
- Reconocimiento e interpretación

2.2.2. Inteligencia Artificial

La inteligencia artificial es la aplicación de procesamiento rápido de datos, aprendizaje automático, análisis predictivo y automatización para simular comportamiento inteligentes y capacidades de resolución de problemas con máquinas y software. Las máquinas y los programas que usan inteligencia artificial generalmente están diseñados para leer e interpretar una entrada de datos y luego responder a ella mediante análisis predictivos o aprendizaje autónomo (The data science and artificial intelligence, 2019).

Un área de aplicación de la inteligencia artificial están los autos autónomos: los sistemas impulsados por inteligencia artificial como estos integran algoritmos de inteligencia artificial, como el aprendizaje automático y el aprendizaje profundo, en entornos complejos que permiten la automatización (Cajamarca, 2019).

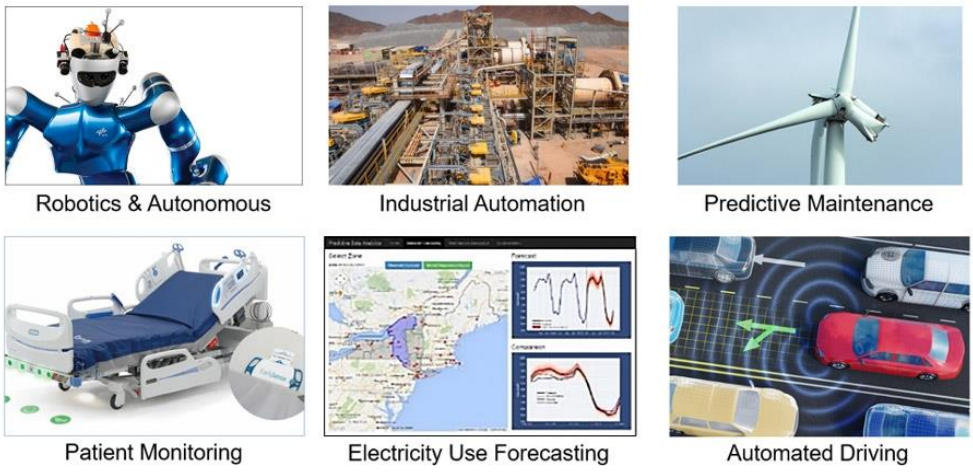


Figura 7. Áreas de aplicación de la Inteligencia Artificial

Fuente: (The MathWorks, 2019)

2.2.3. Aprendizaje Autónomo (Machine Learning)

El aprendizaje autónomo es un campo de la inteligencia artificial que tiene como objetivo enseñar a las computadoras a hacer lo que es natural para los humanos y animales: aprende de la experiencia (MathWorks, 2019). Es por ello que el aprendizaje automático implica la construcción de algoritmos que adaptan sus modelos para mejorar su capacidad de hacer predicciones. Los algoritmos de aprendizaje automático usan métodos computacionales para "aprender" información directamente de los datos sin depender de una ecuación predeterminada como modelo (DeepAI, 2019). Cabe mencionar que los algoritmos mejoran su rendimiento de forma adaptativa a medida que aumenta el número de muestras disponibles para el aprendizaje.

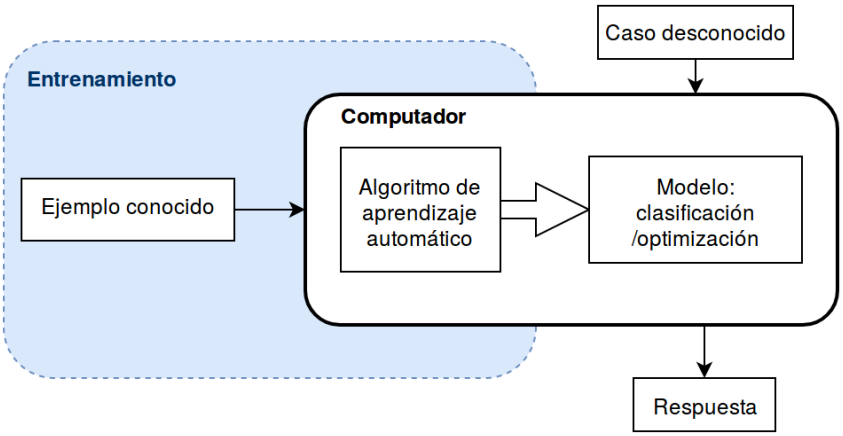


Figura 8. Esquema de Aprendizaje Automático

Fuente: (DeepAI, 2019)

2.2.3.1. Técnicas de Aprendizaje Automático

El aprendizaje automático utiliza dos tipos de técnicas: Aprendizaje supervisado que entrena un modelo en datos de entrada y salida conocidos que puede predecir resultados futuros y aprendizaje no supervisado, que encuentra patrones ocultos o estructuras intrínsecas en los datos de entrada (Fernando Sancho Caparrini, 2020).

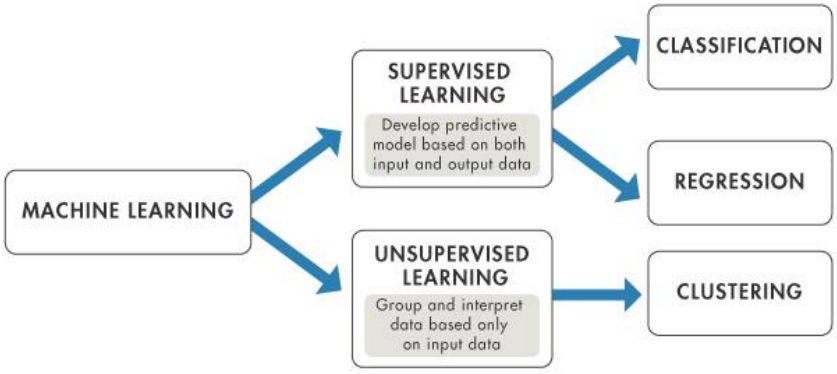


Figura 9. Técnicas de Aprendizaje Automático

Fuente: (MathWorks, 2019)

2.2.4. Aprendizaje Profundo (Deep Learning)

El aprendizaje profundo es una técnica de aprendizaje automático que enseña a las computadoras a hacer lo que es natural para los humanos: aprender con el ejemplo (MathWorks, 2019). De otra forma más puntual puede definirse al aprendizaje profundo como un conjunto de algoritmos de aprendizaje automático que adquieren un aprendizaje de principio a fin donde se le brindan unos datos de entrada (imágenes, texto o sonidos) y una tarea para que lo ejecute el algoritmo y este de forma automáticamente aprende a realizar la respetiva clasificación (Lucas García, 2016).

En la figura 11, se observa por medio de un esquema el comportamiento del flujo de trabajo de aprendizaje automático y aprendizaje profundo.

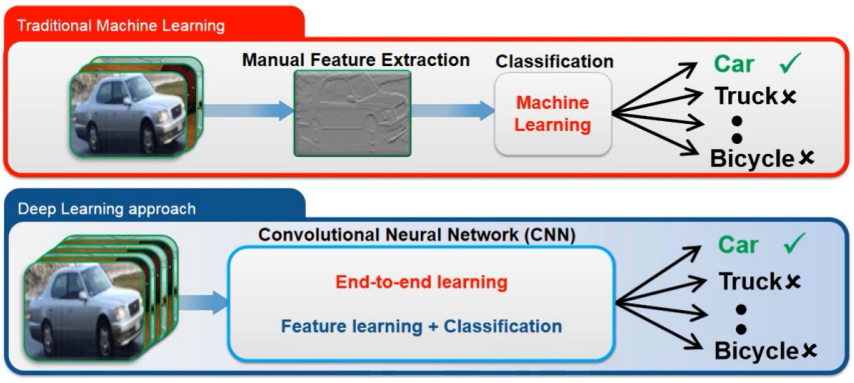


Figura 10. Comparativa de flujo de trabajo entre Aprendizaje autonómico y Aprendizaje profundo

Fuente: (Lucas García, 2016)

Hoy en día no ha empezado a ser útil el aprendizaje profundo, y esto se debe a dos motivos. EL primero, se requiere grandes cantidades de datos; y el segundo motivo, requiere una potencia de cálculo significativa. Las GPU de alto rendimiento tiene una arquitectura paralela que resulta eficiente para el aprendizaje profundo. Algunos usos prácticos del aprendizaje profundo que se pueden tener son (DeepAI, 2019):

- Conducción Autónoma
- Sector aeroespacial y de defensa
- Reconocimiento automático de voz
- Investigación medica
- Automatización industrial

El aprendizaje profundo basa su funcionamiento en el uso de arquitecturas de redes neuronales profundas. El término "profundo" hace referencia al número de capas ocultas en la red neuronal, las redes neuronales convencionales solo contienen 2 o 3 capas ocultas, mientras que las redes profundas pueden tener hasta 150 capas.

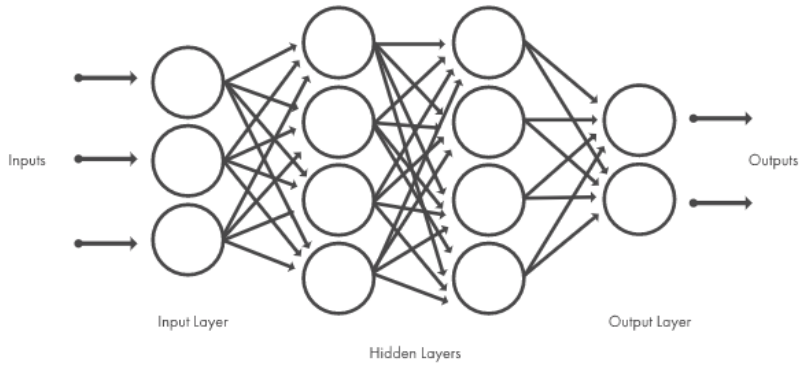


Figura 11. Redes neuronales organizadas en capas

Fuente: (DeepAI, 2019)

2.2.4.1.Redes Neuronales Convolucionales (CNN)

Una red neuronal convolucional es una red multicapa jerárquica comúnmente utilizada para el aprendizaje profundo, además, es un tipo de aprendizaje automático en el que un modelo

aprende a realizar tareas de clasificación, para las aplicaciones de interés se hablará de imágenes (MathWorks, 2019).

Las capas de la red funcionan como filtros de diferentes resoluciones que se le aplican a la imagen convolutiva y que sirven de entrada a la siguiente capa, extraen características de mayor interés, aumentando la complejidad de detalle conforme avanzan en las capas de la red. Este tipo de red tiene una estructura muy similar a la de una red neuronal típica, pero una CNN se tiene tres tipos de capas diferentes que son:

- Una capa convolucional
- Una capa de *pooling* o de reducción, esta capa tiene la tarea de reducir el número de parámetros requeridos por la red, y solo adquiere las características más comunes de las imágenes.
- Una capa clasificadora que está completamente conectada, la misma que va a realizar la clasificación de los objetos y así arrojar los resultados.

A continuación, se muestra en la figura 13 una arquitectura típica de la Red Neuronal Convolucional.

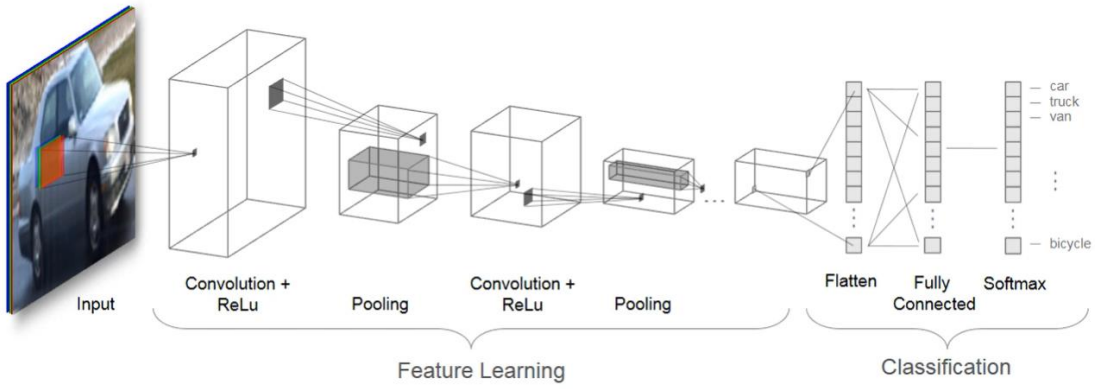


Figura 12. Arquitectura de una red neuronal convolucional

Fuente: (García L. , 2017)

2.2.5. Entrenamiento de una CNN

En esta sección se dará a cabo la explicación de cómo entrenar una CNN. Como se menciona anteriormente existe dos posibilidades en el entrenamiento de una CNN, ya sea desde cero, o utilizando una red previamente entrenada para realizar el entrenamiento con los nuevos datos.

2.2.5.1.Entrenamiento de una CNN desde cero

El entrenamiento de una red neuronal convolucional desde cero es una opción que se elige cuando no se tiene ningún modelo previamente entrenado de la misma red. Además, es necesario tener en cuenta que para problemas de la vida real y un buen funcionamiento la base de datos de imágenes debe ser bastante alto, incluso pueden llegar hacer miles de imágenes por categoría, esto significa un costo computacional bastante alto por lo que se necesita la ayuda de *GPU* para el procesamiento, provocando la reducción del tiempo. Para el uso del software de Matlab, se recomienda el uso de una *GPU NVIDIA* con una capacidad computacional 3.0 o superior (MatWorks, 2016).

Para desarrollar el entrenamiento de una red neuronal convolucional desde cero, es necesario seguir los siguientes pasos que se dan a conocer a continuación:

- Es necesario que todas las imágenes que van a hacer utilizadas en la entrada de la red deben ser homogéneas, es decir, que tengan las mismas características en cuanto al tamaño se refiere.
- Seguido se debe contar con las imágenes etiquetadas, y debido a la cantidad de imágenes necesarias, podría ser una labor importante ya que cada una debe estar etiquetada.
- Se elige una arquitectura para la implementación de la CNN, la misma que dará inicio al entrenamiento y clasificación, la parte que más tiempo y recursos requiere.
- Por último, la red entrenada es sometida a pruebas, la cual se debe hacer un análisis de los resultados obtenidos y en base a ellos elegir si los mismos fueron los que satisfacen a la solución del problema o si por lo contrario es necesario repetir el procedimiento.

2.2.5.2. Entrenamiento de una CNN utilizando una red pre-entrenada (Transfer Learning)

Por lo general, entrenar una CNN cuesta mucho tiempo y ciclos de GPU. Una técnica clave para evitar este tipo de costo es el "Aprendizaje de transferencia", el cual tiene como objetivo aprovechar las características que ya aprendió una CNN y por medio de un pequeño re-entrenamiento de las ultimas capas, se ajuste al reconocimiento de nuevas imágenes (MathWorks, 2018). Las ventajas que muestra este modelo es que el costo computacional es mucho menor y por ende los resultados se obtienen de manera más rápida que realizar un entrenamiento desde cero, cabe mencionar que el rendimiento de la CNN dependerá de la red pre-entrenada que se utilice. En la figura 14 se muestra gráficamente la representación de dicho proceso.

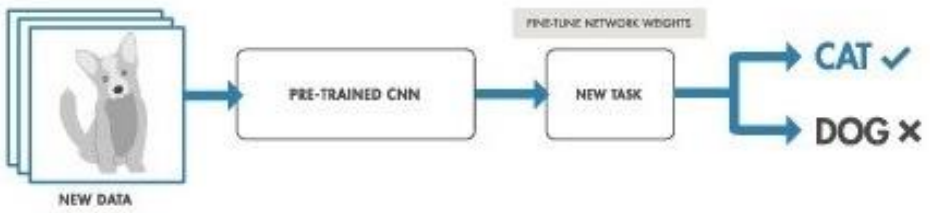


Figura 13. Formación de un modelo por Aprendizaje de transferencia

Fuente: (MathWorks, 2017)

Existen varias redes neuronales convolucionales pre-entrenadas, las mismas que nos pueden ayudar a realizar una clasificación de manera más fácil con tan solo volviendo a reentrenar a la red con los parámetros que mejor se adapten a la solución del problema. A continuación, se dará una breve explicación de las redes convolucionales ya entrenadas como son: VGG16 (Karen Simonyan, 2015), ResNet (Burgal, 2018), GoogleNet (Das V. &., 1017) y una de las redes más comunes que existe, la red neuronal convolucional AlexNet en la cual nos vamos a enfocar debido a que se la utilizó en la presente investigación.

2.2.5.3.AlexNet (2012)

AlexNet es una red neuronal convolucional pionera de las redes neuronales convolucionales. Fue creada en el año de 2012 por Alex Krizhevsky, Ilya Sutskener y Geoffrey Hinton en una competencia que se realiza cada año sobre visión por computadora llamada ILSVRC (ImagenNet Large-Scale Visual Recognition Challenge). La red logro un error de 15.3% más de 10.8 puntos de porcentaje por delante del finalista. La red AlexNet tiene 8 capas de profundidad, y fue entrenada en más de un millón de imágenes de la base de datos ImageNet (ImageNet, 2013). La red pre-entrenada puede clasificar imágenes en 1000 categorías de objetos, como teclado, mouse, lápiz, animales, etc. Como resultado, la red ha aprendido representaciones de características ricas para una amplia gama de imágenes. La red tiene un tamaño de entrada de imagen de 227 por 227 (Alex Krizhevsky, 2012).

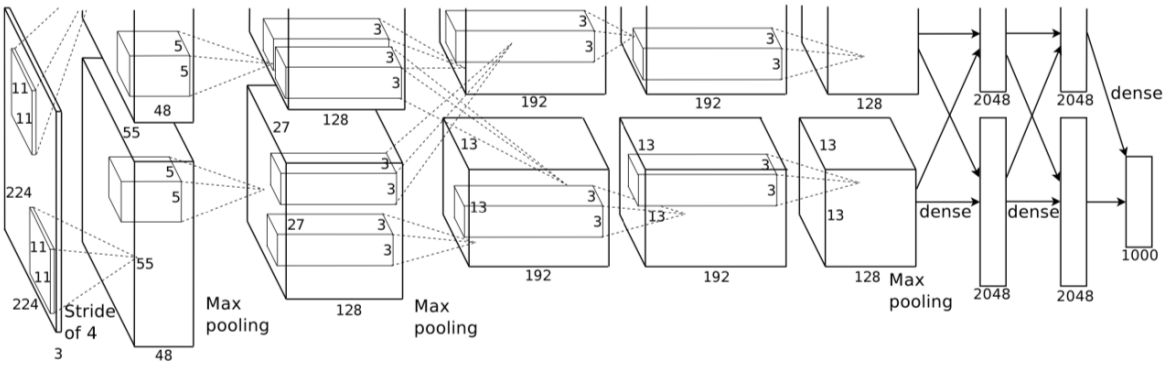


Figura 14. Ilustración de la arquitectura de AlexNet

Fuente: (Alex Krizhevsky, 2012)

El diseño de la red AlexNet, además, incluyo las técnicas de *dropout* y *data augmentation* durante el entrenamiento.

Dropout cosiste en fijar en 0 la salida de cada neurona con una probabilidad de 0.5. Las neuronas que pasan por esta acción no contribuyen al paso hacia adelante por la red. Esto resulta en que, para cada nueva muestra de entrenamiento, la red presenta una arquitectura distinta, pero cada arquitectura contiene los mismos pesos. Esta técnica reduce la co-dependencia entre neuronas, facilitando el aprendizaje de características más robustas.

Data augmentation o comúnmente llamado aumento de datos, corresponde a la técnica de transformar las imágenes de entrenamiento para aumentar el tamaño de la base de datos y prevenir el sobreajuste. Las dos técnicas más comunes para *Data augmentation* son: una es tomar partes aleatorias de 224x224 de las imágenes de entrenamiento; la segunda forma es alterar las intensidades de los canales RGB en las imágenes de entrenamiento.

2.2.6. Detectores basados en CNN

Con el paso de los años los detectores de objetos en las imágenes han aprovechado las virtudes y características de las CNN, debido a los buenos resultados obtenidos en comparación a otras técnicas de identificación. A continuación, se da a conocer las características de los detectores basados en Redes Neuronales Convolucionales (CNN).

2.2.6.1.R-CNN

El R-CNN (Regional CNN) tiene como labor capturar la imagen de entrada y lograr identificar en qué lugar de esta se encuentra el objeto a identificar y este a su vez es resaltado por un cuadro delimitador, es decir, tiene como entrada a las imágenes, y las salidas son los recuadros delimitadores, a este se le agrega la etiqueta del objeto que encontró en la imagen.

Su funcionamiento se basa en proponer un conjunto de cuadros en la imagen (propuestas de región), es un método donde se usa el algoritmo de búsqueda selectiva (Uijling J.R.R, 2013) para extraer solo 2000 regiones de la imagen. Una vez identificadas las regiones propuestas o de interés, el algoritmo procede a crear una deformación de la respectiva imagen, extrayéndola de la región, de tal manera que este adaptada a las dimensiones y sirva para entrada a la CNN en donde cumple el procedimiento ya antes mencionado, seguido a ello en la capa final de la CNN se realiza la etapa de clasificación por medio de un arreglo SVM (Support Vector Machine) que servirá de clasificador y por último se aplica la regresión lineal para obtener la imagen con el recuadro seleccionando el objeto (García B. S., 2018). En la figura 16, se muestra gráficamente las etapas que sigue la metodología del R-CNN para la detección de objetos.

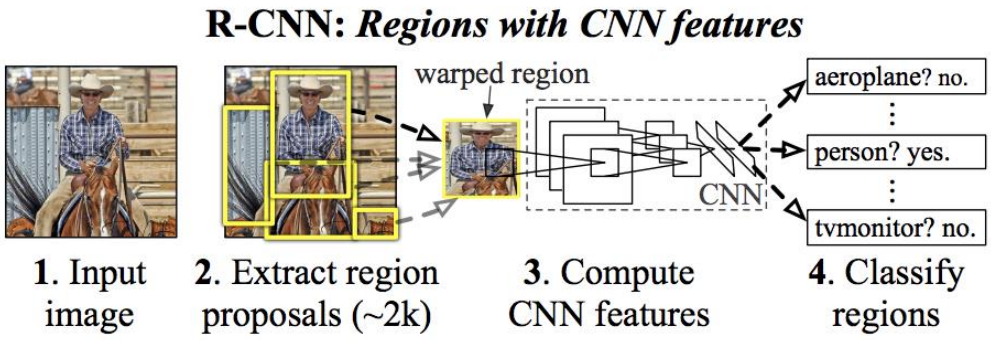


Figura 15. Funcionamiento R-CNN

Fuente: (Shaoqing Ren, 2016)

2.2.6.2.Fast R-CNN

Este modelo de detector corrige y mejora algunos de los aspectos del modelo anterior, para ello, se unificó 3 procedimientos por separado en un marco de entrenamiento conjunto y aumentando los resultados del cálculo compartido. En lugar de extraer vectores de características de la CNN para cada región, este modelo los agrega en la CNN mientras pasa por toda la imagen y las regiones comparten esta matriz de características. Tiene como primera modificación la aplicación de la técnica del *ROI Pooling* (Erdem, 2019), lo cual permite extraer las características de las ROI propuestas de manera externa que a su vez disminuye el tamaño, generando nuevas ROI mucho más sencillas para el procesamiento. La segunda modificación consiste en realizar la extracción, clasificación y regresión en una sola operación de la red reemplazando el *SVM* por una capa de *softmax y de regresión*, para así realizar las predicciones. En la figura 17, se muestra gráficamente las etapas que sigue la metodología del Fast R-CNN para la detección de objetos.

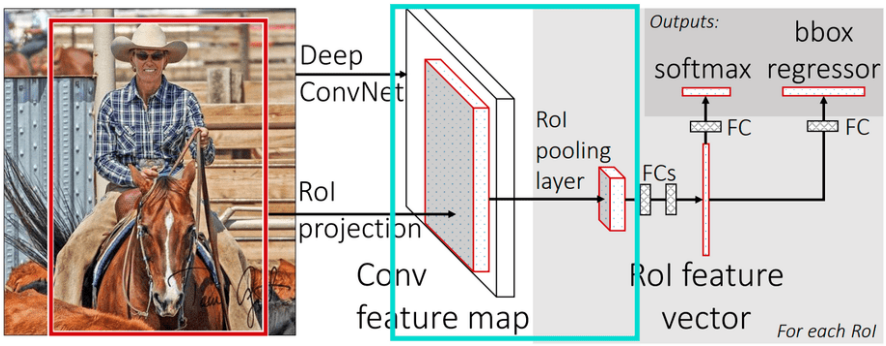


Figura 16. Estructura Fast R-CNN

Fuente: (Girshick, 2015)

2.2.6.3.Faster R-CNN

La estructura Faster R-CNN es en la cual nos vamos a enfocar debido a que se la utilizó en la presente investigación. Esta nueva estructura fue creada por un grupo de desarrollo de Microsoft, corrigiendo un pequeño detalle del funcionamiento del Fast R-CNN, para la anterior estructura es necesario contar con las regiones propuestas como entrada, el problema consiste en que para realizar esas regiones en las imágenes toma un tiempo considerable con metodologías tradicionales, así es que esta nueva estructura, se aprovecha el mapa de características, como salidas de la CNN para generar las regiones propuestas o ROI. En la figura 18, se muestra gráficamente el esquema que sigue la metodología del *Faster R-CNN* para la detección y clasificación de objetos en comparación con los dos anteriores.

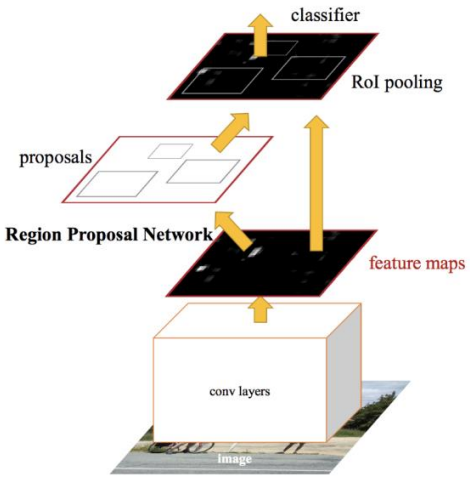


Figura 17. Funcionamiento Faster R-CNN

Fuente: (Shaoqing Ren, 2016)

Su funcionamiento consiste en que una ventana recorre el mapa de características de la CNN, generando “K” cuadros y puntajes que tan probable es que sea una ROI (García B. S., 2018), estos deben ser de las proporciones del tamaño del objeto que se espera encontrar en la imagen; estos cuadros o cajas con tamaños definidos y similares entre si son conocidos como cuadros de anclaje. Una vez definidas las ROI, el proceso es idéntico al del *Fast R-CNN*, con estas nuevas imágenes con las ROI definidas por la RPN como entrada.

2.2.7. Vehículos Utilizados

A lo largo del proyecto se trata sobre videos captados a través de una cámara fpv, que a su vez es movilizado por un Vehículo Aéreo no Tripulado (Drone).

El drone con el cual se comenzaron a estudiar sus videos, y la mayor parte de este proyecto es usando dichos videos. Es el Mavic Pro s2, se trata de un drone tipo quadcopter, el cual tiene la función de ir desplazándose por la superficie, lleva incorporada una cámara de video.

Figura 18. Drone Mavic 2

Fuente: (Autor)

CAPÍTULO III

3. METODOLOGÍA

3.1 MARCO METODOLÓGICO

Para la elaboración del presente trabajo de investigación, la metodología empleada se basó en dos atributos: experimental y explicativo donde se puede identificar los procesos que se utilizó para recolectar y clasificar la información necesaria, de esta manera realizar el presente trabajo.

3.2 TIPO DE ESTUDIO

Investigación de tipo experimental y explicativa, es experimental porque requiere el desarrollo e implantación de un sistema para la detección y clasificación de la contaminación superficial (objetos solidos), a través de escenas o imágenes provenientes de una cámara, además de la utilización de un UAV para el despliegue de la cámara y los sensores bio-ambientales que puedan trabajar en conjunto con el centro de control operado a larga distancia.

Investigación explicativa ya que se van a explicar en qué consisten los algoritmos de inteligencia artificial y tratamiento de imágenes, además de especificar los procesos necesarios para la realización de los diseños propuestos en el presente trabajo.

3.3. MÉTODOS DE INVESTIGACIÓN

Para la realización del presente proyecto se aplicará los métodos analítico y deductivo debido a que realiza un análisis de la generación tecnológica de Inteligencia Artificial y utilización de sistemas de clasificación necesarias para el correcto diseño y desarrollo correspondiente del sistema propuesto.

3.4. FUENTES DE INFORMACIÓN

Se ha realizado una revisión sistemática de documentos de varias sociedades científicas como ResearchGate, IEEE, revistas, publicaciones científicas destacadas, todas dedicadas al análisis de algoritmos de inteligencia artificial. Además de lo antes mencionado se recurrió a páginas web entre ellas el sitio oficial de MathWorks y materiales electrónicos adicionales para el desarrollo del sistema para la detección y clasificación de la contaminación superficial.

3.5. INSTRUMENTOS DE LA INFORMACIÓN

Para realizar este proyecto de investigación los instrumentos utilizados son los siguientes: software de diseño, publicaciones científicas, revistas, papers, libros, datasheets y se empleó la técnica de la observación que consiste en capturar y visualizar de forma detallada, cualquier situación o fenómeno que se produzca, en función a los objetivos de la investigación ya antes planteados.

3.6. POBLACIÓN Y MUESTRA

El presente proyecto de investigación se realizará en la Laguna de Colta en un lugar de preferencia del entorno, como prioridad se tomarán sus orillas.

Las imágenes tomadas por el UAV en toda la periferia de la laguna de Colta serán las muestras que posteriormente serán analizadas mediante los algoritmos de Inteligencia Artificial.

3.7 OPERACIONALIZACIÓN DE LA VARIABLES

Tabla 1. Operacionalización de variables

Fuente: (Autor)

Variable	**Concepto**	**Categoría**	**Indicadores**	**Técnicas e Instrumentación**
Independiente Cantidad de desechos sólidos	Acumulación de cualquier basura, desperdicio y otros materiales sólidos de desechos, resultantes de las actividades domiciliarias, industriales y comerciales.	Medio Ambiente	-Fragmentación del Habitad -Clasificación de residuos -Porcentaje del tipo de material contaminante encontrado -Total de residuos sólidos generados -Número de desechos sólidos no encontrados	-Cámara FPV -Sensores Bio-ambientales -Algoritmos de Inteligencia Artificial
Dependiente Contaminación Ambiental	Es cualquier cambio químico, físico o biológico en la calidad del agua que tiene un efecto dañino en cualquier cosa viva que consuma esa agua.	Medio Ambiente	-Nivel de contaminación -Porcentaje de contaminación hídrica -Emisiones de CO2 por semana	-Sensores Bio-ambientales -Sistema de reconocimiento y clasificación de la contaminación superficial

3.8. PROCEDIMIENTOS Y ANÁLISIS

En esta sección se describe el desarrollo del sistema para reconocer y clasificar contaminación superficial en la Laguna de Colta utilizando algoritmos de inteligencia artificial.

El proceso general para la implementación de un sistema para reconocer y clasificar contaminación superficial en la imagen de salida de la cámara FPV se muestra en la figura 20. En esta sección se describe en detalle cada paso del proceso.

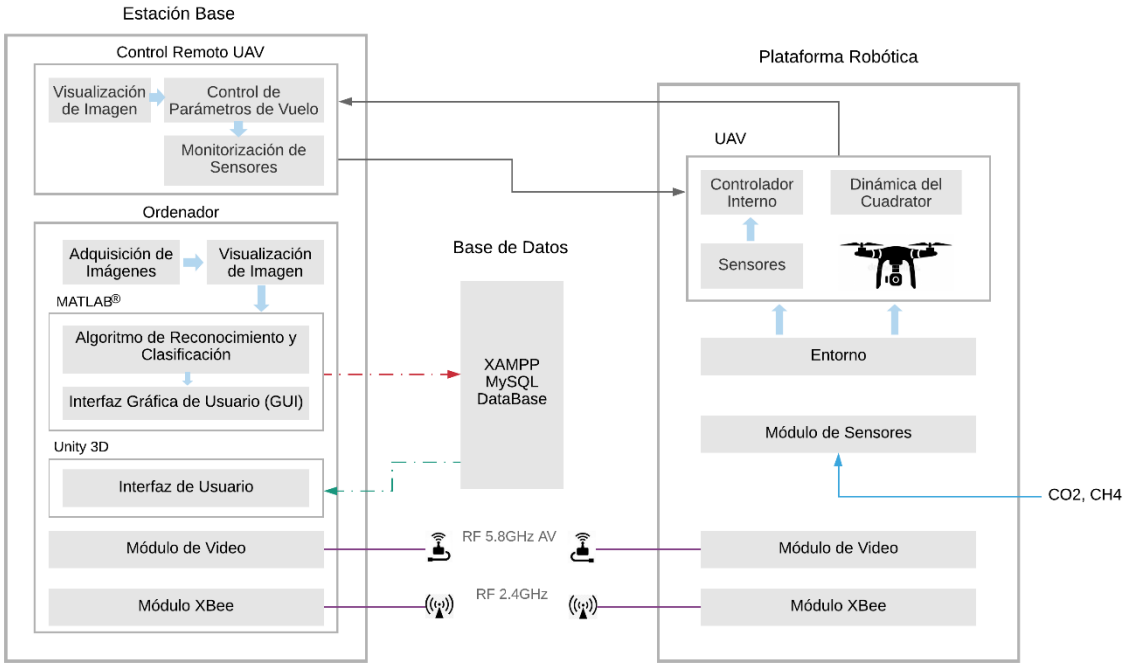


Figura 19. Diagrama general del sistema para el reconocimiento y clasificación de contaminación superficial

Fuente: (Autor)

3.8.1. ALGORITMO PARA EL RECONOCIMIENTO Y CLASIFICACIÓN

En este apartado, se describirá en detalle toda la metodología desarrollada para la clasificación de objetos en fotografías aéreas captadas por la cámara FPV, incluida la relación de las características de los datos y la elección de parámetros. Para ello se va a utilizar la red neuronal *AlexNet* la cual se volverá a reentrenar para que pueda clasificar únicamente los objetos de interés del proyecto, así como el desarrollo de la Interfaz Gráfica en los softwares utilizados Matlab y Unity para llevar a cabo la visualización de los registros.

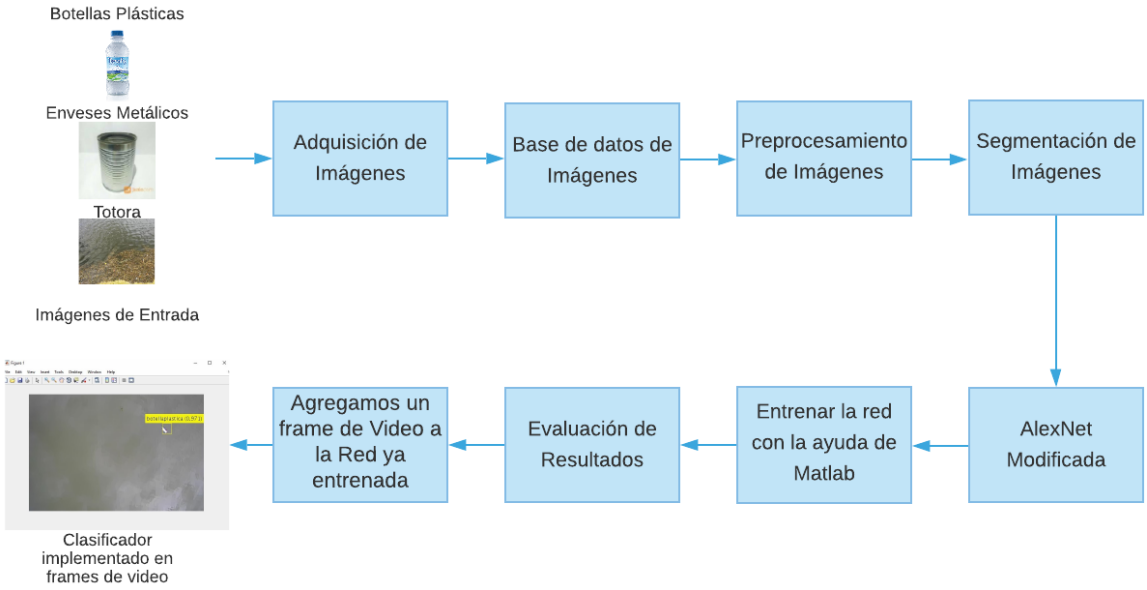


Figura 20. Representación gráfica del algoritmo para reconocer y clasificar la contaminación superficial

Fuente: (Autor)

3.8.2. CRITERIOS DE DISEÑO

Para este trabajo se definirán 3 categorías que generalmente se encuentran en la superficie de la Laguna de Colta: botellas plásticas, envases metálicos y totora, para cada categoría se utilizó un banco de imágenes de entre 900 y 1200 imágenes.

Antes de seguir con el desarrollo del trabajo se mostrarán los requisitos necesarios o capacidad computacional adecuado para el entrenamiento de la red neuronal convolucional

ya que al momento de entrenar la red neuronal se requiere mayor acceso de memoria y mayor procesamiento.

3.8.2.1.Recursos Utilizados

Para la realización de este proyecto se utilizaron los componentes que se describen brevemente a continuación:

3.8.2.1.1. Hardware

- Una máquina de alta tecnología para poder utilizar procesamiento paralelo con las siguientes características mínimas que se describen a continuación:
  Acer Aspire E15 - Portátil Full HD de 15,6
  - CPU: Intel(R) Core(TM) i7-6500U 7ª generación 2.5 – 3.1 GHZ
  - Memoria: RAM 8GB DDR4
  - Disco Duro: 1TB HDD
  - Sistema Operativo: 64 bits
- (Huawei P9 Lite), Cámara 13 megapíxeles con flash led integrada.

3.8.2.1.2. Software

Para poder integrar el software es necesario poder contar con:

- Matlab y sus respectivas herramientas que hacen parte directamente de la realización del programa correspondiente al presente trabajo (es necesario contar con la versión 2017b del programa o superiores).
- Unity 3D el cual servirá para mostrar los resultados al usuario de los registros del sistema que están en la base de datos.
- XAMPP o hosteo en la nube que debe contar con el servicio del gestor de base de datos MySQL, que a su vez se almacenaran los registros del sistema.

3.8.2.2.Instalación de Toolbox Model for AlexNet Network

Para la instalación de dicho paquete se siguieron los siguientes pasos:

El primer paso a realizar es ingresar al MATLAB, una vez en el programa vamos a la pestaña *HOME* y podemos agregar capacidades adicionales con dar clic en la opción *Get Add-Ons* como se muestra en la Figura 22.

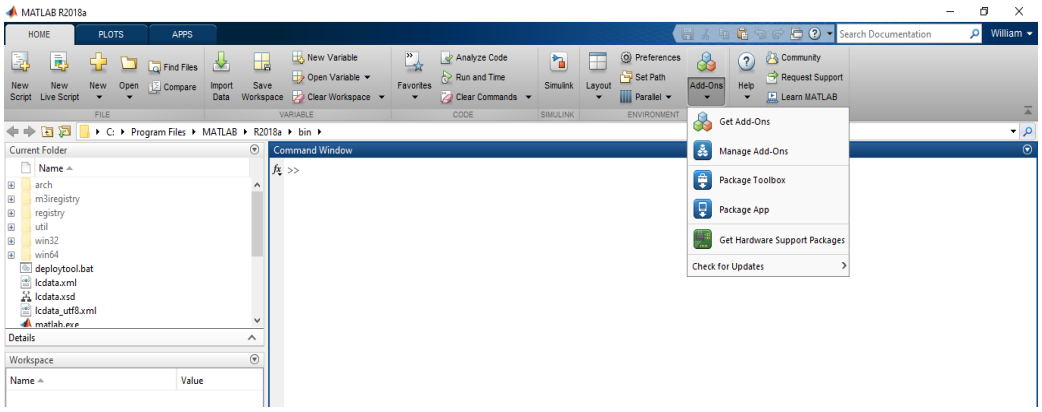


Figura 21. Pantalla principal con icono de complementos de MATLAB

Fuente: (Autor)

Una vez ya en la pantalla principal, permitirá agregar herramientas de inteligencia artificial que se requieran, en este caso vamos a buscar el nombre de AlexNet ToolBox e instalar como se observa en la Figura 23.

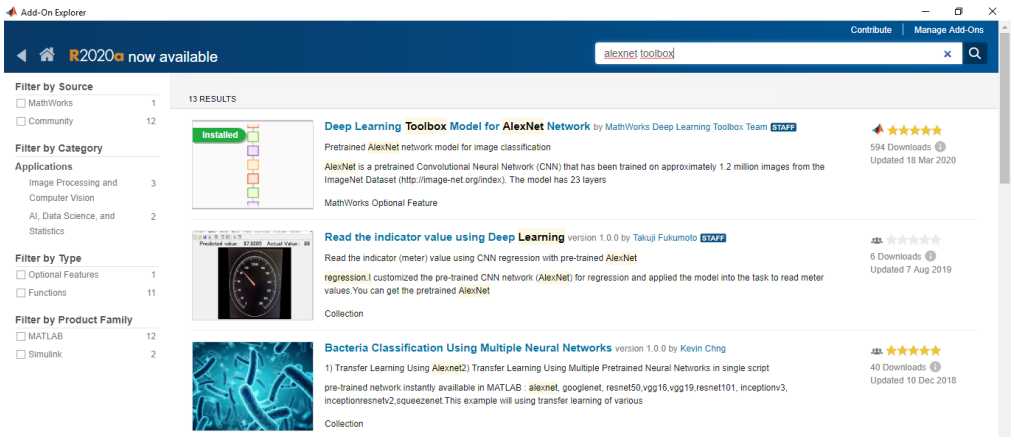


Figura 22. Pantalla de Add-Ons, del ToolBox for AlexNet ya instalada

Fuente: (Autor)

Si la red neuronal ya se encuentra instalada, la forma de saber que está funcionando en nuestro sistema, es dirigirnos a la ventana de comandos y escribir la palabra “alexnet” y así nos mostrara datos de la CNN como se muestra en la Figura 24.

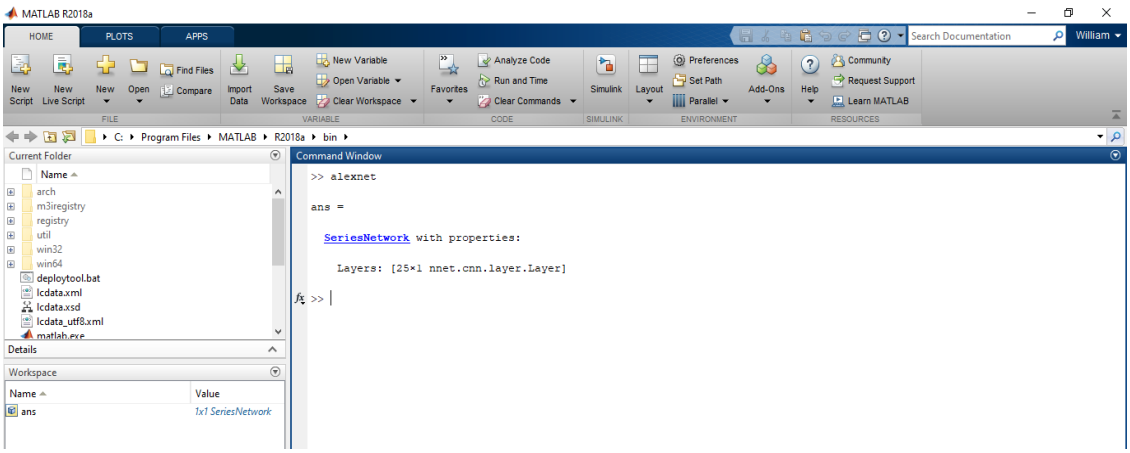


Figura 23. Verificación de la CNN AlexNet en Matlab

Fuente: (Autor)

Con la red instalada y funcionando en nuestro sistema, se procede a realizar la modificación de la red neuronal para que pueda clasificar los objetos de interés del trabajo a desarrollarse. Por la cual se muestra una secuencia de procesos realizados en este proyecto para la adaptación y reentrenamiento de la CNN AlexNet con el clasificador de tipo Faster R-CNN.

3.8.3. DESARROLLO DEL PROCESO DE ENTRENAMIENTO DE LA RED DE TIPO FASTER R-CNN

En este apartado para realizar una explicación de manera concreta del proceso que se realizó en la modificación y reentrenamiento de la CNN de tipo faster R-CNN (Regiones con redes neuronales convolucionales), se creó un diagrama de bloques el cual explica rigurosamente el proceso de cada fase tal como se muestra en la Figura 33.

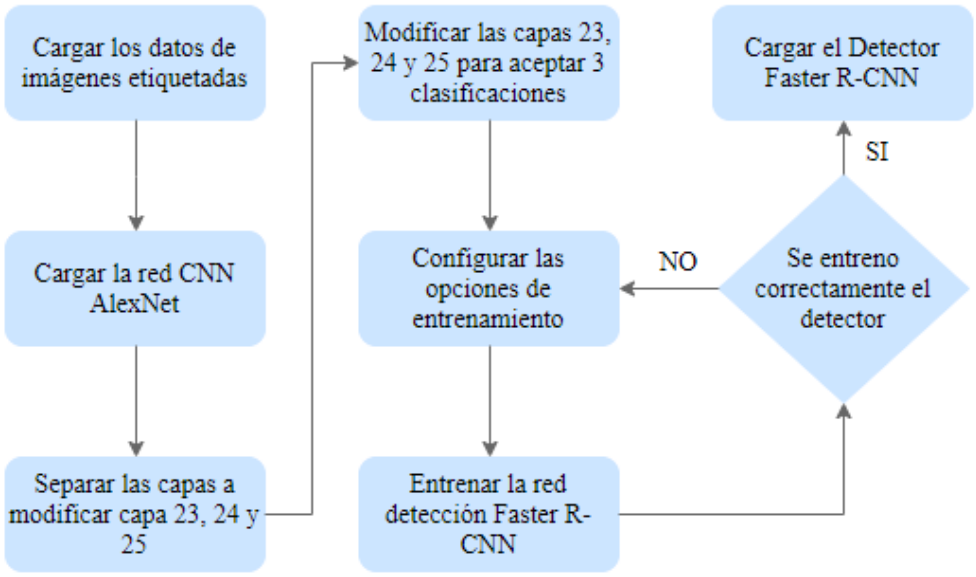


Figura 24. Proceso de entrenamiento de la red

Fuente: (Autor)

Este es el esquema en el cual se trabajaron los distintos entrenamientos para la creación de los diferentes detectores de objetos contaminantes que se describen más adelante, además las configuraciones con mayor eficiencia servirán para conseguir el entrenamiento deseado que sea capaz de detectar y clasificar las 3 clases de contaminación superficial presentes en la laguna.

Cabe mencionar que el detector sobre el cual se va a trabajar está basado en ROI (región rectangular de interés) de tipo *faster* RCNN (MathWorks, 2020), el mismo que necesita una base de datos de las imágenes etiquetadas donde se le indiquen los objetos de interés y su ROI delimitadas con un recuadro. Para crear una base de datos de imágenes con sus respectivas etiquetas se usa una aplicación de etiquetado de objetos y funciones de *Computer Vision Toolbox* para entrenar algoritmos a partir de los datos. Los pasos por seguir para crear datos de entrenamiento para detección de objetos se pueden apreciar en la siguiente Figura 26.

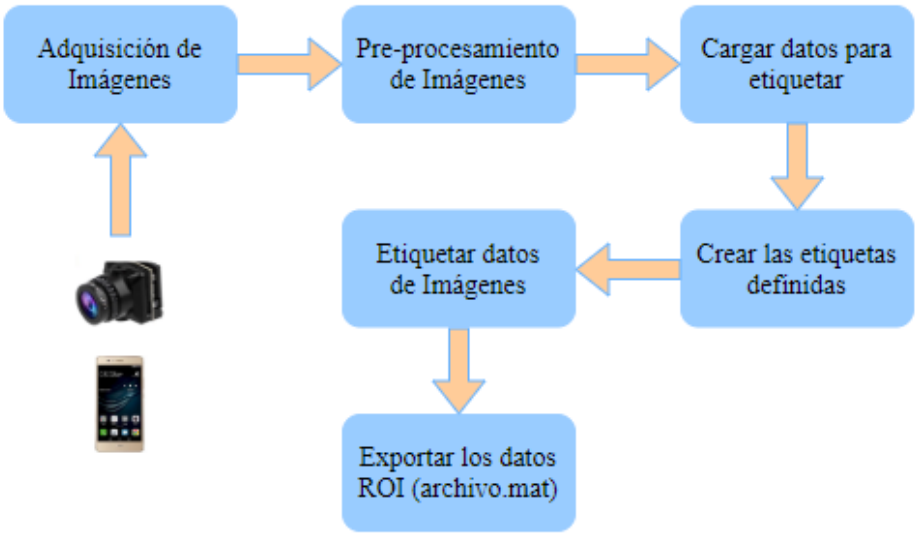


Figura 25. Crear datos de Entrenamiento

Fuente: (Autor)

3.8.3.1.Adquisición de Imágenes

La adquisición de las respectivas imágenes de los objetos contaminantes (botellas plásticas, envases metálicos y totora), se las realiza utilizando una cámara fpv cuyas especificaciones se detallan en el anexo 2 y un Smartphone Huawei p9 Lite con una cámara de 13 Mpx.

Por facilidad, tiempo y capacidades técnicas, se elige un tramo de patio al exterior de la vivienda donde se procederá a la toma de imágenes que contengan los objetos en estudio, cabe mencionar que para la toma de imágenes de la clase totora nos dirigiremos a la Laguna de Colta.

Para la adquisición de las imágenes utilizando una cámara fpv se lo realiza por medio de un sistema de video inalámbrico a una frecuencia de 5.8 Ghz y con la ayuda de Matlab el cual proporciona una línea de código como muestra la figura 27, para la adquisición de las imágenes.

```
%% Código para adquirir imágenes
% Construir un objeto de entrada de video.
vidObj = videoinput('winvideo', 1, 'UYVY_720x480');
src = getselectedsource(vidObj);
triggerconfig(vidObj ,'manual');
set(vidObj, 'TriggerRepeat',inf);
set(vidObj, 'FramesPerTrigger',1);
vidObj.FrameGrabInterval = 1;
vidObj.ReturnedColorspace = 'rgb';
vidObj.ROIPosition = [50 30 620 420];
start(vidObj);
pause(4);
%% Funcion para tomar la foto
for p=1:1
    foto=getsnapshot(vidObj);              % Adquiere un solo cuadro de imagen.
    pause(0.5);                            % Es el tiempo de espera
    imwrite(foto,strcat(num2str(p),'.jpg'));% nombra las fotos en formato jpg
end
%%
stop(vidObj),clear vidObj % detiene y limpia el objeto de video
%delete(vid);             % elimina el objeto de entrada de video de la memoria.
```

Figura 26. Código para adquirir imágenes

Fuente: (Autor)

El proceso de recopilación de imágenes inicia con la captura de los objetos de interés con la cámara especificada, a una altura de 1.5m aproximadamente entre los objetos y la cámara, y teniendo muy en cuenta evitar que exista contraluz con la cámara y reflexión con el objeto, esto se realiza con el propósito de obtener una cantidad suficiente de imágenes para lograr expresar la mayor cantidad de detalles disponibles. Estas imágenes conformaran la base de datos para el desarrollo del sistema.

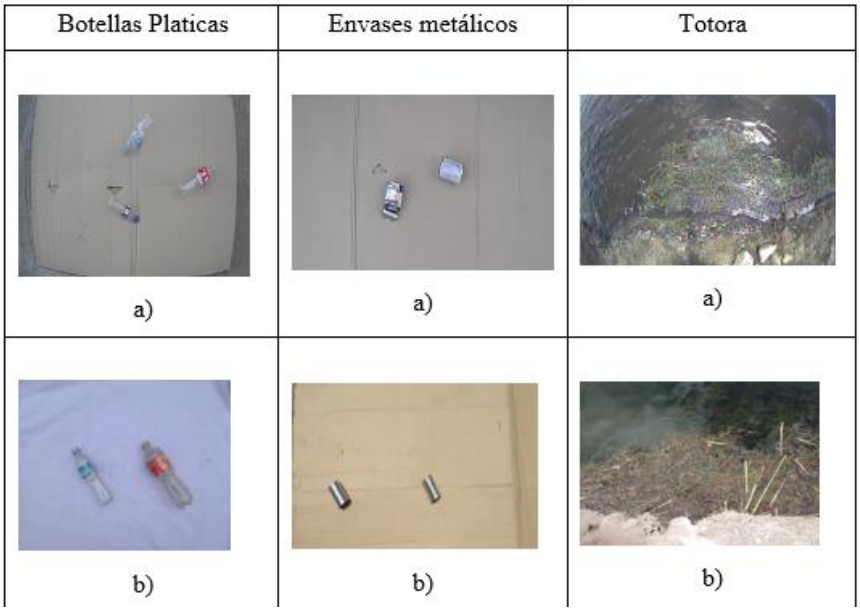


Figura 27. Ilustración de muestras para cada clase a) cámara fpv, b) cámara del celular

Fuente: (Autor)

3.8.3.2.Pre-procesamiento de Imágenes

Las imágenes que se obtuvieron de los objetos puestos en estudio se tendrán que redimensionar debido a que la red *AlexNet* en su primera capa necesita una imagen de entrada 227x227x3. Además, como ya se mencionó anteriormente, un problema muy común es el no poder contar con la cantidad de imágenes suficientes a la hora de entrena a la red, por tal inconveniente se utiliza el concepto de *Data Augmentation*, esto nos permite aumentar la cantidad de imágenes por cada objeto contaminante.

Para el redimensionamiento y las rotaciones de las imágenes se realiza con el código siguiente tal como muestra en la Figura 29.

```
%% Redimencionar y rotar las imágenes
% Elegir las imágenes del Directorio en formato .JPG
Folder = 'C:\Users\rickl\Documents\Aumentarimagenes';
a = dir(fullfile(Folder, '*.jpg'));

for k = 1:length (a)
    i = imread (a(k).name);
    % Redimencionar las imágenes a 227x227
    img = imresize(i, [227 227]);

    img2 = imrotate (img, 90);  % Rotar las imágenes
    img3 = imrotate (img, 180); % Rotar las imágenes
    img4 = imrotate (img, 270); % Rotar las imágenes
    % Guardar las imágenes
    imwrite(img, sprintf('botella_plastica%i.jpg', k));
    imwrite(img2, sprintf('botella_plastica90_%i.jpg', k));
    imwrite(img3, sprintf('botella_plastica180_%i.jpg', k));
    imwrite(img4, sprintf('botella_plastica270_%i.jpg', k));
end
```

Figura 28. Código para redimensionar y rotar

Fuente: (Autor)

3.8.3.3.Etiquetado de Imágenes

Luego del procesamiento de las imágenes de botellas plásticas, envases metálicos y totora, se procede a almacenar las fotografías en el computador y se crea carpetas para cada grupo de clases u objetos como se muestra en la siguiente Figura 38.

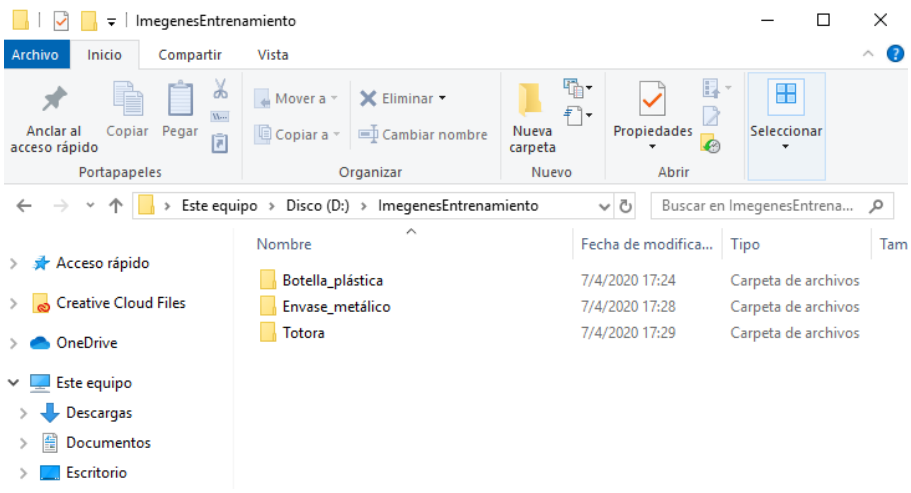


Figura 29. Nombre de la carpeta de imágenes a etiquetar

Fuente: (Autor)

Para crear un base de datos desde el inicio, se necesitará para ello etiquetar las imágenes con las cuales se trabajará con sus respectivas características. Existen varios factores que inciden en su elección tal como es el tamaño de las imágenes, además entre sus herramientas de trabajo no se dispone con un GPU compatible con el software Matlab, por lo que es importante tener en cuenta que la capacidad de procesamiento solo se ejecutara más que por la CPU del computador, además basados a los trabajos de investigación referentes a los detectores y clasificadores, se tiene la iniciativa de trabajar con imágenes redimensionadas a 350 x 300 pixeles como primera posibilidad.

En el caso de Matlab, para poder realizar dicho trabajo tiene una aplicación de etiquetado *Image Labeler*. La aplicación mencionada pertenece a las herramientas que se utiliza para el procesamiento de imágenes y visión artificial como se puede observar en la Figura 31.

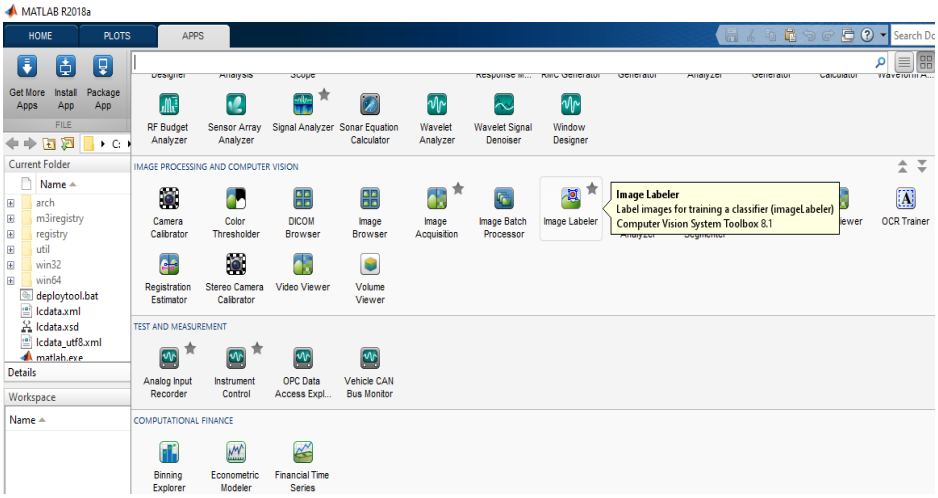


Figura 30. Pantalla de las APPS para el procesamiento de imágenes y visión artificial

Fuente: (Autor)

En primera instancia se ingresa al software Matlab y en la ventana de comando se selecciona la barra de *APPS* y se ingresa a la aplicación *Image Labeler* o su vez ejecutando el comando *imageLabeler* en la ventana de comandos de Matlab que ejecutará la herramienta, tal como se observa en la Figura 31.

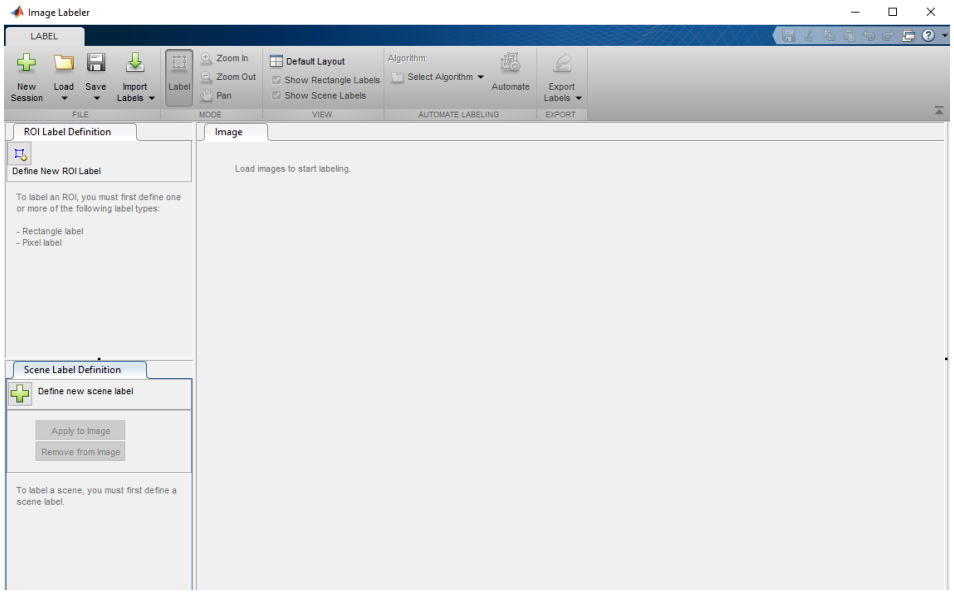


Figura 31. Entorno de APP Image Labeler

Fuente: (Autor)

En la figura 31 se puede observar las funciones que posee esta herramienta para su funcionamiento, lo primero a realizar es cargar las carpetas con sus respectivas imágenes a etiquetar, una vez cargadas las imágenes se crean las respectivas etiquetas a utilizar (botella plástica, envase metálico y totora) en el etiquetado, para lo cual se procede a la realización del etiquetado de las imágenes. Esta es una tarea bastante intensiva y que consume mucho tiempo. En las imágenes 32,33 y 34 se muestra el proceso de etiquetado de las imágenes con sus respectivas clases que fueron elegidos por el sistema a clasificar.

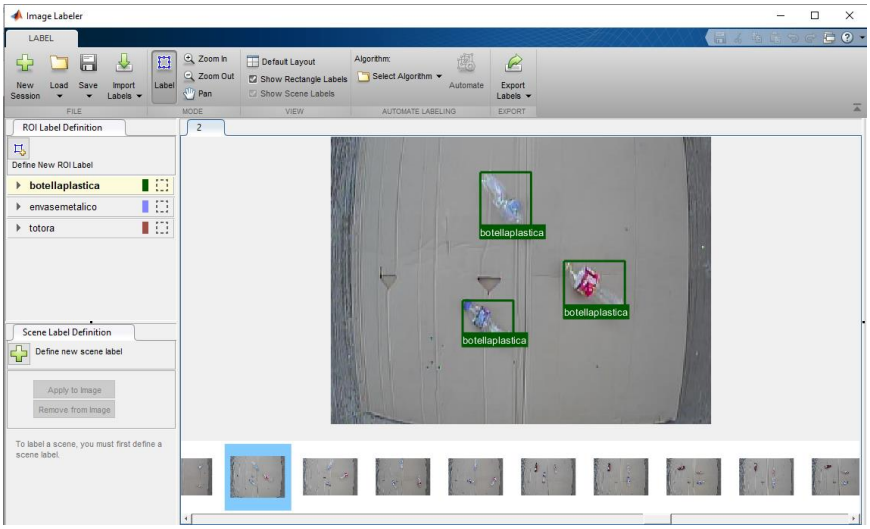


Figura 32. Etiqueta de Botellas Plásticas

Fuente: (Autor)

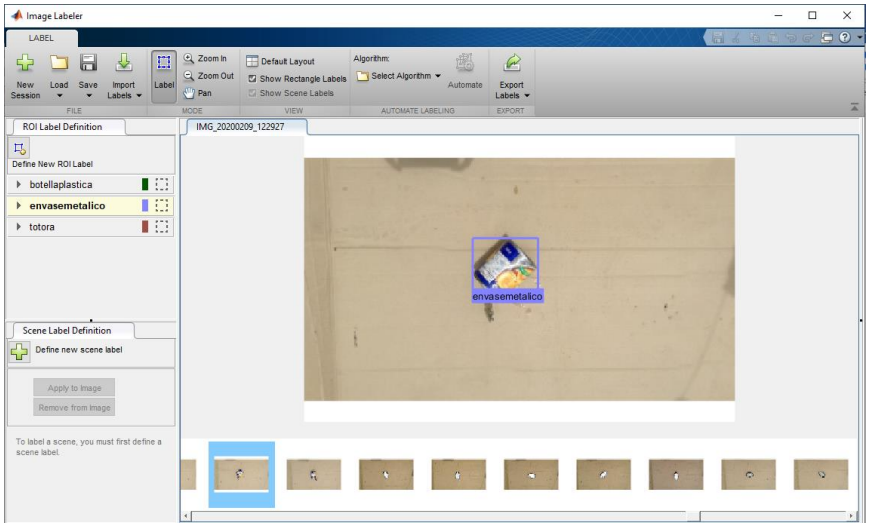


Figura 33. Etiqueta de Envases Metálicos

Fuente: (Autor)

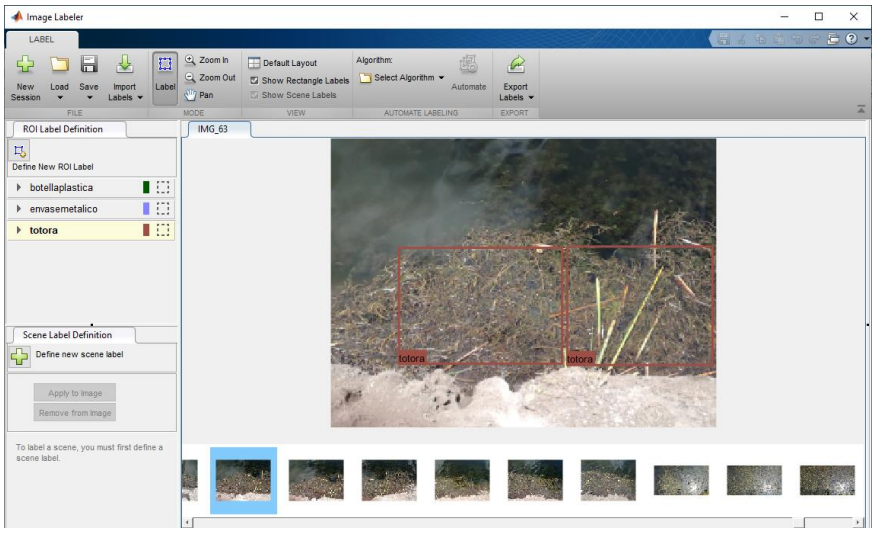


Figura 34. Etiqueta de Totora

Fuente: (Autor)

Una vez realizado todo el etiquetado, se procede a exportar las etiquetas el cual creará un fichero (*archivo.mat*) para así poder cargarlas en el *Workspace* de Matlab y realizar el entrenamiento del detector.

3.8.3.4. Creación de script del detector y clasificador de tipo Faster R-CNN en Matlab

Con los parámetros y requerimientos disponibles se procede al desarrollo del código del detector, que para ello se tomó como referencia el código estructural para detectores entrenados por MathWorks.

Como se mencionó anteriormente, el modelo propuesto inicial debe ser modificado y entrenado tantas veces sean necesarios hasta obtener el mejor resultado para la detección y clasificación. En la tabla 2 se encuentran los valores establecidos.

Tabla 2. Configuraciones de entrenamiento iniciales

Fuente: (Autor)

Configuraciones de entrenamiento					
Configuración	Cantidad de Imágenes	Tamaño de Imágenes	Tamaño del filtro	Filtros primera capa	Taza Inicial de Aprendizaje
1	356 300	227 227 3 227 227 3	[11 11]	96	0.001

3.8.3.5.Entrenar detector objeto contaminante 1 (Botellas Plásticas)

Cargar el conjunto de datos

Se ingresa a Matlab, una vez ahí se crea un script con el nombre FRCNNbotellaplastica.m dicho script contendrá todo el código de entrenamiento de la CNN.

Se procede a cargar el archivo *LabelBotellaPlastica.mat* de las imágenes con todas las etiquetas, para ello, se utiliza la función *'objectDetectorTrainingData'* que a la vez crea una variable ('botellaplasticaDataset') que contiene los datos extraídos del archivo anterior.

```
%% Entrena un Faster RCNN detecor de objetos contamimates(botellasplasticas, env

%% Cargar los datos de entrenamineto
load('D:\TesisRCNN\RCNN_botellaplastica\LabelBotellaPlastica.mat');
botellaplasticaDataset = objectDetectorTrainingData(gTruth,'SamplingFactor',1);
```

Figura 35. Realizar un Dataset

Fuente: (Autor)

En la figura 36, entre los resultados se observa la variable ('botellaplasticaDataset') que almacena 356 imágenes con su respectiva etiqueta en forma de tabla.

De la base de datos se divide en dos conjuntos uno para entrenamiento para entrenar el detector y otro conjunto de prueba para evaluar el detector. Se toma el 70% del total de los

datos para el entrenamiento y el 30% se utiliza para la evaluación siendo las recomendaciones de Mathworks.

```
%%
idx = floor(0.7 * height(botellaplasticaDataset));
trainingData = botellaplasticaDataset (1:idx,:);
testData = botellaplasticaDataset (idx:end,:);
```

Figura 36. Dividir los datos en un conjunto de entrenamiento y prueba

Fuente: (Autor)

Cargar la red neuronal convolucional (AlexNet)

La red AlexNet es la cual será modificada para clasificar solo 3 clases de interés de nuestro sistema. Se procede a cargar la red pre-entrenada AlexNet, proceso sencillo debido a su integración con Matlab. La carga de AlexNet se llevó a cabo con solo almacenar en un variable *net*, que va a tener de extensión *convnet*, tal como se muestra en la figura 38, y además se puede revisar la arquitectura de la red con la ayuda del comando *net.Layers* tal como se muestra en la figura 39.

```
%% Cargar la red pre-entrenada (AlexNet)
net = alexnet;
%revisar la arquitectura de red
layers = net.Layers
```

Figura 37. Cargar la red AlexNet

Fuente: (Autor)

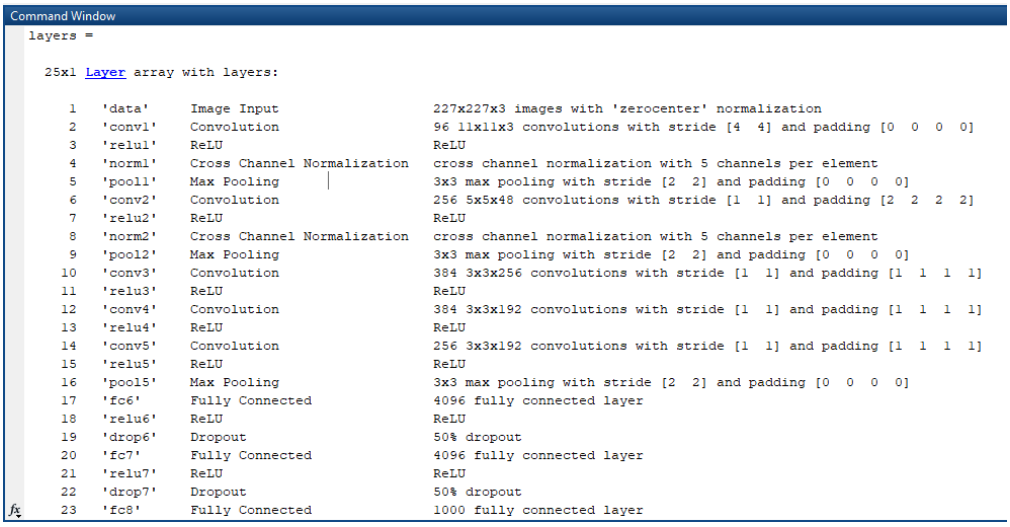


Figura 38. Arquitectura de la red AlexNet

Fuente: (Autor)

Como se observa en la Figura 39, *AlexNet* es una red pre-entrenada de 23 capas. La capa 21 es una capa totalmente conectada que tiene como salida 1000 categorías de imágenes. Seguido se especifica el número de categorías a clasificar, en este caso 3 categorías (botellas platicas, envases metálicos y totora), por lo que se tiene que modificar las tres últimas capas que se explicaran a en la siguiente sección.

Modificar las capas de la red

La red en la última capa realiza la clasificación de 1000 objetos, para que la red se adapte a nuestro clasificador se tiene que modificar las tres últimas capas para tener como salida únicamente tres categorías.

Las capas tienen como función definir la arquitectura de la red y contienen los pesos previamente aprendidos, por tal motivo no se modificarán las capas iniciales, pero si las tres últimas capas que son las de clasificación, todo el proceso de modificación se observa en la Figura 40.

```
% Modificar la red pre-entrenada
num_objects = size(botellaplasticaDataset,2)-1; %numero de categorias
layers(23) = fullyConnectedLayer(num_objects, 'Name','fc8');
layers(25) = classificationLayer('Name','myNewClassifier')

layers(end-2).WeightLearnRateFactor = 40;
layers(end-2).BiasLearnRateFactor = 40;
```

Figura 39. Modificar las tres últimas capas

Fuente: (Autor)

En la Figura 40, se observa cómo fueron modificadas las tres últimas capas de la red: `FullyConnectedLayer`, `SoftmaxLayer` y `ClassificationLayer` con los parámetros para que clasifique solo tres categorías de objetos (botella plástica, envase metálico y totora), teniendo en cuenta que los pesos de las capas inferiores no van a cambiar y solo se cambiara los pesos de las capas superiores para su mayor aprendizaje.

Configurar opciones de entrenamiento

El detector de tipo *Faster RCNN* entrena el detector en cuatro etapas, cada una de las cuatro etapas de entrenamiento usa su propio conjunto de opciones de entrenamiento, establecidas por la función *'trainingOptions'* en cada elemento y guardadas en la variable '*optionsStageX*', siendo X el número de la etapa.

```
%%
%Options for step 1.
optionsStage1 = trainingOptions('sgdm',...
    'MaxEpochs',10,...
    'MiniBatchSize', 128,...
    'InitialLearnRate', 1e-3,...
    'CheckpointPath', tempdir);
%Options for step 2.
optionsStage2 = trainingOptions('sgdm',...
    'MaxEpochs',10,...
    'MiniBatchSize', 64,...
    'InitialLearnRate', 1e-3,...
    'CheckpointPath', tempdir);
%Options for step 3.
optionsStage3 = trainingOptions('sgdm',...
    'MaxEpochs',10,...
    'MiniBatchSize', 128,...
    'InitialLearnRate', 1e-3,...
    'CheckpointPath', tempdir);
%Options for step 4.
optionsStage4 = trainingOptions('sgdm',...
    'MaxEpochs',10,...
    'MiniBatchSize', 64,...
    'InitialLearnRate', 1e-3,...
    'CheckpointPath', tempdir);
```

Figura 40. Características de las etapas de entrenamiento

Fuente: (Autor)

Una vez entrenada cada etapa, se agrupan todas las 'optionsStage' en un vector llamado 'options' tal como se puede apreciar en la siguiente Figura 42.

```
options = [
    optionsStage1
    optionsStage2
    optionsStage3
    optionsStage4
    ];
```

Figura 41. Array de opciones de cada etapa

Fuente: (Autor)

Entrenar el detector de objetos Faster R-CNN

Con la CNN y las opciones de entrenamiento ya configuradas, utilizamos el comando 'trainFasterRCNNObjectDetector' para empezar a entrenar el detector.

Los valores de entrenamiento que se utilizó para el detector fueron: NegativeOverlaRange [0 0.3], PositiveOverlapRange [0.6 1], SmallestImageDimension, 350, BoxPyramidScale, 1.2.

Al entrenar un detector de tipo faster R-CNN, tendrá como componentes de entrada la tabla 'trainingData', la CNN, en este caso AlexNet, y el vector options. Con todo lo mencionado se forma el detector.

```
rng(0);
tic;
detectorbotellaplastica = trainFasterRCNNObjectDetector(plasticDataset, net, options, ...
        'NegativeOverlapRange', [0 0.3], ...
        'PositiveOverlapRange', [0.6 1], ...%'SmallestImageDimension', [400], ...
        'SmallestImageDimension', [350], ...
        'BoxPyramidScale', 1.2);
    toc;
```

Figura 42. Crear el detector

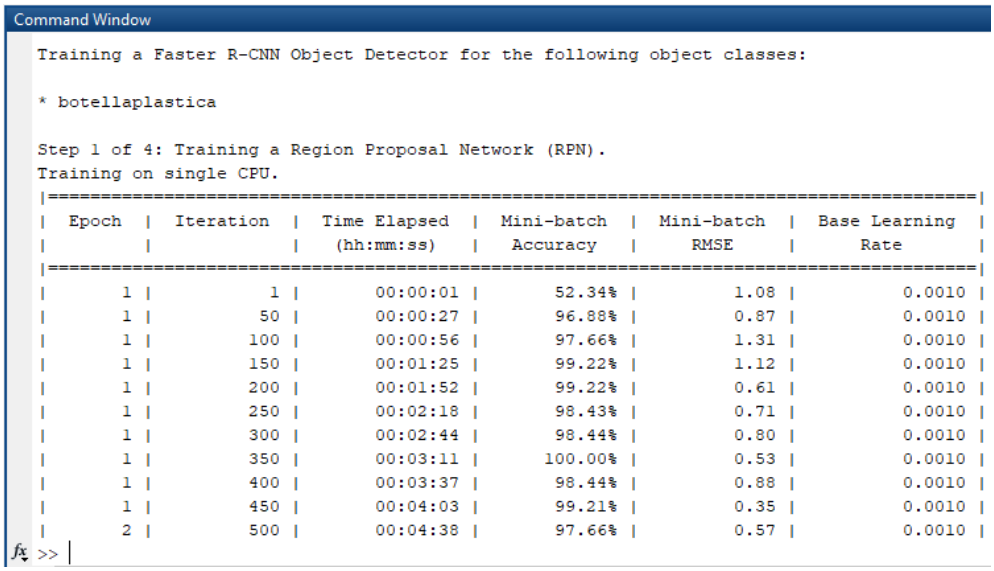
Fuente: (Autor)

Ya con los parámetros anteriores especificados, solo queda indicar los valores del Parallel Computing Toolbox. Para realizar el procesamiento en paralelo solo basta con ir a la parte inferior izquierda del Matlab y hacer clic en la opción de *Start Parellel Pool* y automáticamente el *parallel pool* se iniciará con el perfil que esta por defecto.

Se procede a la etapa de crear el clasificador y posteriormente a la inicialización del entrenamiento. Como se ha mencionado, es un entrenamiento que implica mucho tiempo. En una primera prueba, con el etiquetado de un objeto contaminante y teniendo unas 700 imágenes en un ordenador portátil tardaba alrededor de 3 horas.

Tabla 3. Datos de entrenamiento botellas plásticas 1

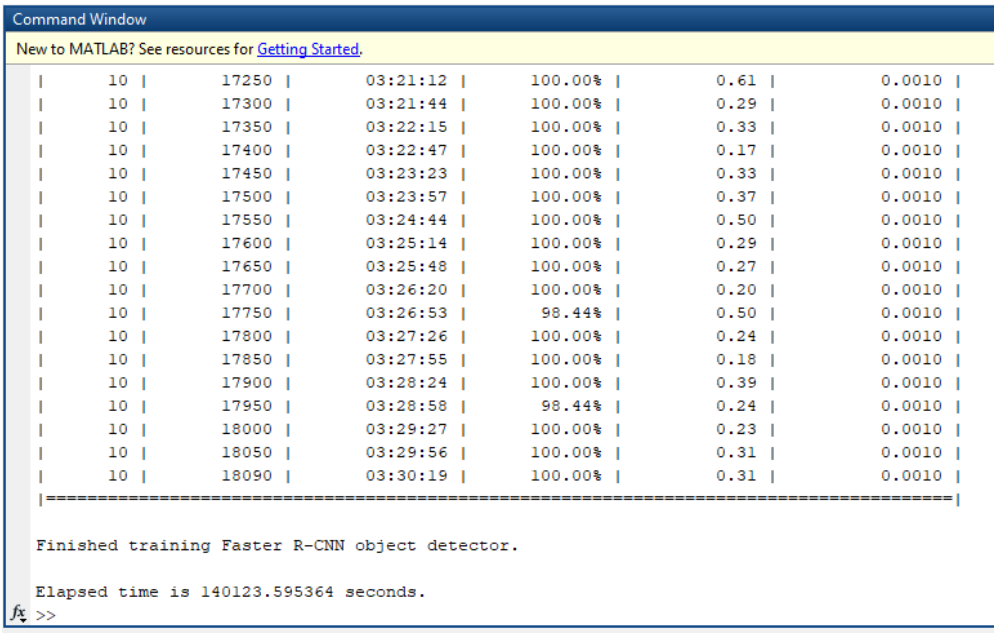
Fuente: (Autor)



Command Window

```
Training a Faster R-CNN Object Detector for the following object classes:

* botellaplastica

Step 1 of 4: Training a Region Proposal Network (RPN).
Training on single CPU.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     RMSE     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:01 |       52.34% |         1.08 |          0.0010 |
|       1 |          50 |       00:00:27 |       96.88% |         0.87 |          0.0010 |
|       1 |         100 |       00:00:56 |       97.66% |         1.31 |          0.0010 |
|       1 |         150 |       00:01:25 |       99.22% |         1.12 |          0.0010 |
|       1 |         200 |       00:01:52 |       99.22% |         0.61 |          0.0010 |
|       1 |         250 |       00:02:18 |       98.43% |         0.71 |          0.0010 |
|       1 |         300 |       00:02:44 |       98.44% |         0.80 |          0.0010 |
|       1 |         350 |       00:03:11 |      100.00% |         0.53 |          0.0010 |
|       1 |         400 |       00:03:37 |       98.44% |         0.88 |          0.0010 |
|       1 |         450 |       00:04:03 |       99.21% |         0.35 |          0.0010 |
|       2 |         500 |       00:04:38 |       97.66% |         0.57 |          0.0010 |
fx >>
```

En la tabla 3, se puede observar los datos generados en el entrenamiento tales como: las iteraciones realizadas por la red, el tiempo que tarda en realizarlas según la configuración de la visualización, la pérdida y precisión de los mini-lotes y la tasa de aprendizaje a lo largo del entrenamiento.



Command Window

New to MATLAB? See resources for Getting Started.

```
|       10 |        17250 |       03:21:12 |      100.00% |         0.61 |          0.0010 |
|       10 |        17300 |       03:21:44 |      100.00% |         0.29 |          0.0010 |
|       10 |        17350 |       03:22:15 |      100.00% |         0.33 |          0.0010 |
|       10 |        17400 |       03:22:47 |      100.00% |         0.17 |          0.0010 |
|       10 |        17450 |       03:23:23 |      100.00% |         0.33 |          0.0010 |
|       10 |        17500 |       03:23:57 |      100.00% |         0.37 |          0.0010 |
|       10 |        17550 |       03:24:44 |      100.00% |         0.50 |          0.0010 |
|       10 |        17600 |       03:25:14 |      100.00% |         0.29 |          0.0010 |
|       10 |        17650 |       03:25:48 |      100.00% |         0.27 |          0.0010 |
|       10 |        17700 |       03:26:20 |      100.00% |         0.20 |          0.0010 |
|       10 |        17750 |       03:26:53 |       98.44% |         0.50 |          0.0010 |
|       10 |        17800 |       03:27:26 |      100.00% |         0.24 |          0.0010 |
|       10 |        17850 |       03:27:55 |      100.00% |         0.18 |          0.0010 |
|       10 |        17900 |       03:28:24 |      100.00% |         0.39 |          0.0010 |
|       10 |        17950 |       03:28:58 |       98.44% |         0.24 |          0.0010 |
|       10 |        18000 |       03:29:27 |      100.00% |         0.23 |          0.0010 |
|       10 |        18050 |       03:29:56 |      100.00% |         0.31 |          0.0010 |
|       10 |        18090 |       03:30:19 |      100.00% |         0.31 |          0.0010 |
|=========================================================================================|

Finished training Faster R-CNN object detector.

Elapsed time is 140123.595364 seconds.
fx >>
```

Figura 43. Fin del entrenamiento

Fuente: (Autor)

Una vez finalizado el entrenamiento del detector, procedemos a guardar el detector, para ello se lo realiza con la ayuda del comando *sabe*, el cual creará una variable de nombre FasterRCNNbotellaplastica, dicho proceso se muestra en la figura 47.

```
%% Guardar el detector una vez finalizado el entrenamineto
save ('D:\TesisRCNN\RCNN_botellaplastica\FasterRCNNbotellaplastica.mat','detectorbotellaplastica')
```

Figura 44. Guardar el detector

Fuente: (Autor)

Evaluar el detector utilizando el conjunto de prueba

En primera instancia el detector debe ser cargado nuevamente con el comando *load* y el nombre con el que se guardó. Para evaluar el detector, se recomienda probarlo en un conjunto de imágenes. Computer Vision System Toolbox™ proporciona funciones de evaluación de detectores para medir métricas comunes con la precisión promedio y la tasa de fallas de promedio de registro. Para la evaluación de nuestro detector se utiliza la métrica de precisión promedio, el código para dicho proceso se puede apreciar en la siguiente figura.

```
%% Evaluar el detector utilizando el conjunto de prueba
% Run detector on each image in the test set and collect results.
    resultsStruct = struct([]);
    for i = 1:height(testData)

        % Read the image.
        I = imread(testData.imageFilename{i});

        % Run the detector.
        [bboxes, scores, labels] = detect(detectorbotellaplastica, I);

        % Collect the results.
        resultsStruct(i).Boxes = bboxes;
        resultsStruct(i).Scores = scores;
        resultsStruct(i).Labels = labels;
    end

    % Convert the results into a table.
    results = struct2table(resultsStruct);

% Extract expected bounding box locations from test data.
expectedResults = testData(:, 2:end);

% Evaluate the object detector using Average Precision metric.
[ap, recall, precision] = evaluateDetectionPrecision(results, expectedResults);
```

Figura 45. Código para evaluar el detector en un conjunto de prueba

Fuente: (Autor)

Una vez obtenidos los valores del detector en un conjunto de prueba se grafica la curva de precisión / recuperación (PR) la cual resalta la precisión de un detector a diferentes niveles de recuperación. Idealmente, la precisión sería 1 en todos los niveles de recuperación, en nuestro primer detector de botellas plásticas, la precisión promedio es ~ 0.6, tal como se puede apreciar en la figura 49.

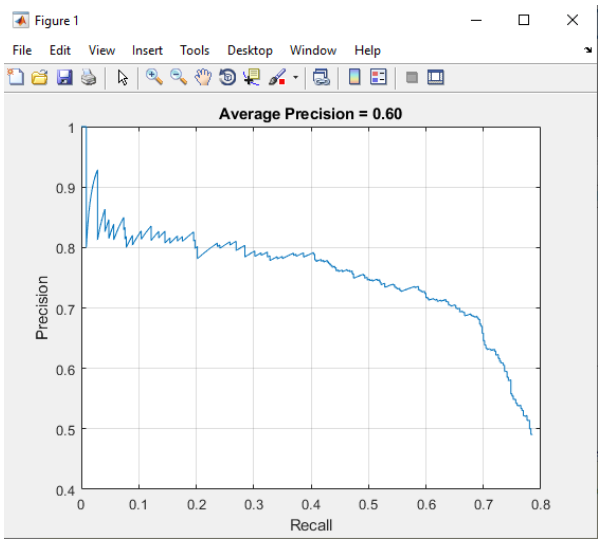


Figura 46. Curva de precisión / recuperación (PR)

Fuente:(Autor)

Probar el detector recientemente capacitado en imágenes de prueba

Para verificar rápidamente la eficiencia del entrenamiento, se ejecuta el detector en una imagen de prueba, la misma que fue tomada del conjunto de imágenes de la base de datos que se utilizó para el entrenamiento de la red.

Para llevar a cabo lo antes mencionado como primer paso es leer una imagen, posterior a ello se procede a ejecutar el detector y como último paso se inserta los cuadros delimitadores seguido de los puntajes de confianza. A continuación, se muestran algunos de los mejores resultados del entrenamiento de la red y se pueden observar en la siguiente figura 51.

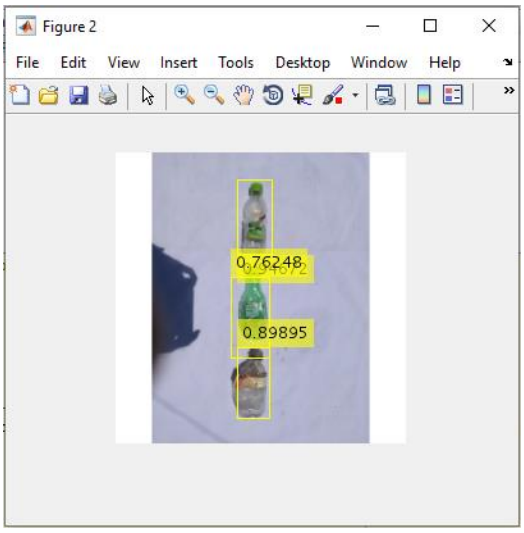


Figura 47. Detección de botella plástica no empleada en entrenamiento, #35

Fuente: (Autor)

En vista a los resultados obtenidos de la primera red entrenada para detectar y clasificar botellas plásticas, surge la necesidad de realizar las diferentes modificaciones de la red hasta obtener una red que arroje los resultados deseados.

Tabla 4. Precisión promedio para botellas plásticas

Fuente: (Autor)

Configuración	Numero de Imágenes	Capa de Entrada - Dimensiones	Tamaño del Filtro	Filtros primera capa	Filtros segunda capa	FC	Épocas	Taza de aprendizaje Inicial	AP
1	356	227	[11 11]	96	256	4026	10	1e-3	0.52
2	356	227	[11 11]	96	256	4026	10	1e-3	0.57
3	356	227	[11 11]	96	256	4026	10	1e-3	0.61
4	650	300	[11 11]	96	256	4026	10	1e-3	0.67
5	650	300	[11 11]	96	256	4026	10	1e-3	0.70
6	750	350	[11 11]	96	256	4026	10	1e-3	0.73
7	750	350	[11 11]	96	256	4026	10	1e-3	0.77
8	950	375	[11 11]	96	256	4026	10	1e-3	0.80
9	950	375	[9 9]	96	256	4026	20	1e-3	0.83
10	1035	400	[9 9]	96	256	4026	20	1e-3	0.86
11	1035	400	[9 9]	96	256	4026	20	1e-3	0.91

3.8.3.6.Entrenar detector objeto contaminante 2 (Envases metálicos)

Para crear el segundo script que contendrá todo el código del detector y clasificador de envases metálicos, se seguirá el mismo procedimiento tal cual se realizó al momento de entrenar el detector del primer objeto contaminante.

Las configuraciones de entrenamiento se realizan tanto en la base de datos como en la red neuronal para poder llegar al resultado deseado. En la tabla 5, se encuentran los valores establecidos con los que se empezara a entrenar la red, aquellos volares son los que ha tenido cambios y los que no se encuentran es porque no han sido modificados.

Tabla 5. Valores de entrenamiento de la CNN

Fuente: (Autor)

Configuraciones de entrenamiento					
Configuración	Cantidad de Imágenes	Tamaño de Imágenes	Tamaño del filtro	Filtros primera capa	Taza Inicial de Aprendizaje
1	378 250	227 227 3 227 227 3	[11 11]	96	0.001

El script que contendrá todo el código para el entrenamiento del detector de envases metálicos está definido con el nombre *FRCNNenvasemetalico.m.*

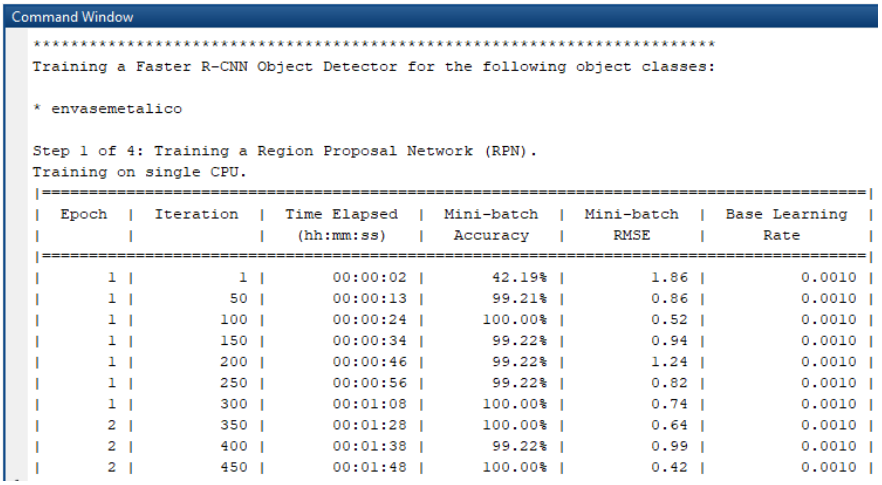
Se procede a la carga del conjunto de datos de 378 imágenes redimensionadas a 227x227 pixeles, seguido se carga la red pre-entrenada Alexnet, la misma que en su primera capa convolucional tendrá un filtro de tamaño [11 11] y una dimensión de 96 filtros, dicho aquello será lo único que se modifique ya que se debe tener en cuenta que los pesos de las capas inferiores no van a cambiar y solo se cambiara los pesos de las capas superiores para su mayor aprendizaje.

Con la CNN y las opciones de entrenamiento ya configuradas, utilizamos el comando ‘trainFasterRCNNObjectDetector’ para empezar a entrenar el detector. Como se ha mencionado, es un entrenamiento lento. En una primera prueba, con el etiquetado de 758

imágenes en el ordenador portátil con las características antes mencionadas y con la cual se está trabajando tarda alrededor de 4 horas.

Tabla 6. Datos de entrenamiento envases metálicos 1

Fuente: (Autor)



Command Window

```
********************************************************************************
Training a Faster R-CNN Object Detector for the following object classes:

* envasemetalico

Step 1 of 4: Training a Region Proposal Network (RPN).
Training on single CPU.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     RMSE     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:02 |       42.19% |         1.86 |          0.0010 |
|       1 |          50 |       00:00:13 |       99.21% |         0.86 |          0.0010 |
|       1 |         100 |       00:00:24 |      100.00% |         0.52 |          0.0010 |
|       1 |         150 |       00:00:34 |       99.22% |         0.94 |          0.0010 |
|       1 |         200 |       00:00:46 |       99.22% |         1.24 |          0.0010 |
|       1 |         250 |       00:00:56 |       99.22% |         0.82 |          0.0010 |
|       1 |         300 |       00:01:08 |      100.00% |         0.74 |          0.0010 |
|       2 |         350 |       00:01:28 |      100.00% |         0.64 |          0.0010 |
|       2 |         400 |       00:01:38 |       99.22% |         0.99 |          0.0010 |
|       2 |         450 |       00:01:48 |      100.00% |         0.42 |          0.0010 |
```

En la tabla 6, se puede observar los datos generados en el entrenamiento tales como: las iteraciones realizadas por la red, el tiempo que tarda en realizarlas según la configuración de la visualización, la pérdida y precisión de los mini-lotes y la tasa de aprendizaje a lo largo del entrenamiento.

Una vez que el detector finaliza su entrenamiento, se recomienda probarlo en un conjunto de imágenes (*testData)*, para la evaluación del detector se utiliza la métrica de precisión promedio, idealmente, la precisión sería 1 en todos los niveles de recuperación, en nuestro primer detector de envases metálicos, la precisión promedio es ~ 0.59. tal como se puede apreciar en la Figura 52.

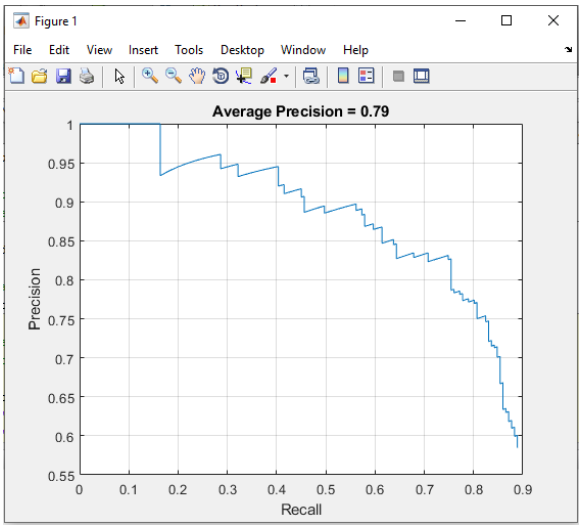


Figura 48. Curva de precisión / recuperación (PR)

Fuente: (Autor)

Para verificar rápidamente la eficiencia del entrenamiento, se ejecuta el detector en una imagen de prueba, la misma que fue tomada del conjunto de imágenes de la base de datos que se utilizó para el entrenamiento de la red.

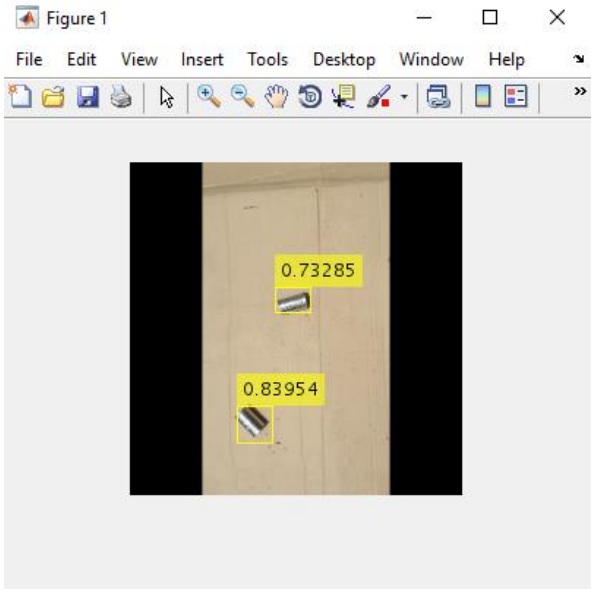


Figura 49. Detección de envase metálico no empleada en entrenamiento, #115

Fuente: (Autor)

El resultado obtenido de la primera red entrenada para detectar y clasificar envases metálicos no cumple con lo esperado, por tal motivo surge la necesidad de realizar las diferentes modificaciones de la red hasta obtener una red que arroje los resultados deseados.

Tabla 7. Precisión promedio para objetos metálicos

Fuente: (Autor)

Configuración	Numero de Imágenes	Capa de Entrada - Dimensiones	Tamaño del Filtro	Filtros primera capa	Filtros segunda capa	FC	Épocas	Taza de aprendizaje Inicial	AP
1	356	227	[11 11]	96	256	4026	10	1e-3	0.43
2	356	227	[11 11]	96	256	4026	10	1e-3	0.49
3	356	227	[11 11]	96	256	4026	10	1e-3	0.54
4	650	300	[11 11]	96	256	4026	10	1e-3	0.59
5	650	300	[11 11]	96	256	4026	10	1e-3	0.62
6	750	350	[11 11]	96	256	4026	10	1e-3	0.66
7	750	350	[11 11]	96	256	4026	10	1e-3	0.79
8	950	375	[11 11]	96	256	4026	10	1e-3	0.80
9	950	375	[9 9]	96	256	4026	20	1e-3	0.83
10	1035	400	[9 9]	96	256	4026	20	1e-3	0.87
11	1035	400	[9 9]	96	256	4026	20	1e-3	0.89

3.8.3.7.Entrenar detector objeto contaminante 3 (Totora)

Para crear el tercer script que contendrá todo el código del detector y clasificador de totora, seguirá el mismo procedimiento tal cual se realizó al momento de entrenar el detector de los objetos contaminantes antes mencionados.

Las configuraciones para el entrenamiento se realizan tanto en el conjunto de imágenes de entrenamiento como en la red neuronal para así poder llegar al resultado deseado. En la tabla 8 se encuentran los valores establecidos con los que se empezara a entrenar la red, aquellos volares son los que ha tenido cambios y los demás no tendrán ningún cambio.

Tabla 8. Valores de entrenamiento de la CNN

Fuente: (Autor)

Configuraciones de entrenamiento					
Configuración	Cantidad de Imágenes	Tamaño de Imágenes	Tamaño del filtro	Filtros primera capa	Taza Inicial de Aprendizaje
1	248 250	227 227 3 300 225 3	[11 11]	96	0.001

El siguiente script denominado *FRCNNtotora.m.* contiene todo el código para el entrenamiento del detector de totora, este a la vez será modificado según los parámetros o valores establecidos por la tabla 8.

Se procede a cargar el conjunto de datos de 385 imágenes redimensionadas a 248x250 pixeles, debido a que los residuos de totora no tienen una forma específica y es por ello que las imágenes contendrán del 100% un 80% totora, además toman un tamaño diferente con respecto a los otros dos objetos contaminantes.

Se procede a cargar la red pre-entrenada AlexNet, a la cual se le realizaron los siguientes cambios: En su primera capa convolucional se decide trabajar con un filtro de tamaño [11 11] y una dimensión de 96 filtros, dicho aquello será lo único que se modifique ya que se debe tener en cuenta que los pesos de las capas inferiores no van a cambiar y solo se cambiara los pesos de las capas superiores para su mayor aprendizaje.

La tabla 9, muestra el desarrollo y estadísticas del entrenamiento del detectar totora.

Tabla 9. Datos de entrenamiento totora 1

Fuente: (Autor)

```
Command Window
********************************************************************************
Training a Faster R-CNN Object Detector for the following object classes:

* totora

Step 1 of 4: Training a Region Proposal Network (RPN).
Training on single CPU.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     RMSE     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:00 |       38.71% |         1.07 |          0.0010 |
|       1 |          50 |       00:00:11 |       96.83% |         0.82 |          0.0010 |
|       1 |         100 |       00:00:22 |       97.62% |         1.22 |          0.0010 |
|       1 |         150 |       00:00:34 |       95.16% |         0.88 |          0.0010 |
|       2 |         200 |       00:00:51 |       92.74% |         0.65 |          0.0010 |
|       2 |         250 |       00:01:02 |       96.06% |         0.78 |          0.0010 |
|       2 |         300 |       00:01:13 |       98.44% |         1.13 |          0.0010 |
|       3 |         350 |       00:01:30 |       94.44% |         0.92 |          0.0010 |
|       3 |         400 |       00:01:41 |       95.28% |         0.69 |          0.0010 |
|       3 |         450 |       00:01:53 |       96.06% |         0.57 |          0.0010 |
|       4 |         500 |       00:02:10 |       96.83% |         0.93 |          0.0010 |
fx >>
```

Luego de finalizar el entrenamiento la red está lista para ser evaluada sobre un conjunto de imágenes (*testData*), para ello se utiliza la métrica de precisión promedio, idealmente, la precisión sería 1 en todos los niveles de recuperación, en nuestro primer detector de totora, la precisión promedio es ~ 0.32. tal como se puede apreciar en la Figura 62.

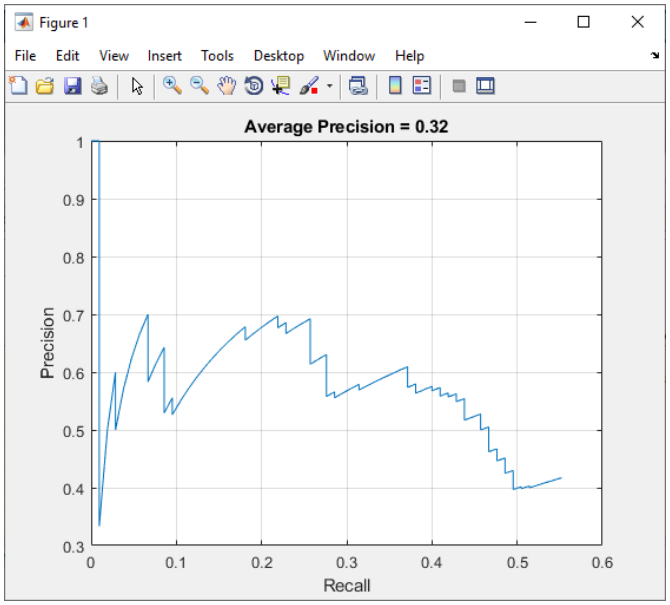


Figura 50. Curva de precisión / recuperación (PR)

Fuente: (Autor)

Ahora probamos el rendimiento del primer entrenamiento del detector totora, el mismo que se realiza ejecutando el detector en una imagen de prueba, dicha imagen es una de las que forman el conjunto de imágenes de prueba (*testData*).

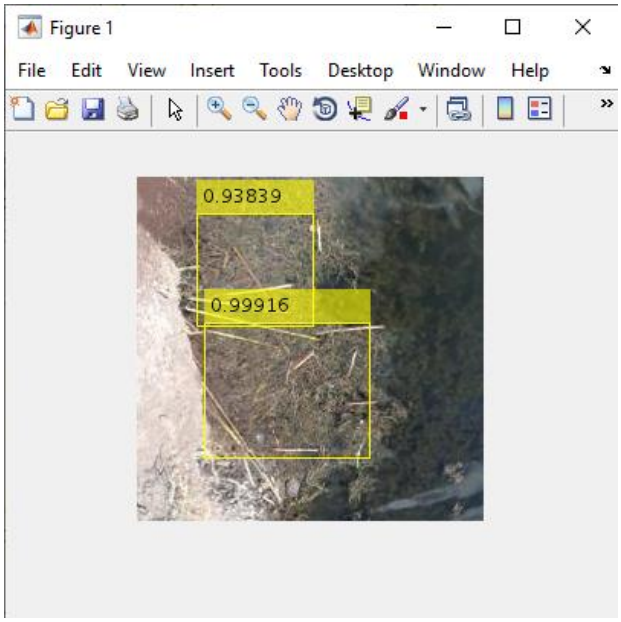


Figura 51. Detección de botella plástica no empleada en entrenamiento, #42

Fuente: (Autor)

El resultado obtenido tras el primer entrenamiento de la red para detectar y clasificar totora no cumple con lo esperado, por tal motivo surge la necesidad de realizar las diferentes modificaciones de la red hasta obtener una red que arroje los resultados deseados.

Tabla 10. Precisión promedio para totora

Fuente: (Autor)

Configuración	Numero de Imágenes	Capa de Entrada - Dimensiones	Tamaño del Filtro	Filtros primera capa	Filtros segunda capa	FC	Épocas	Taza de aprendizaje Inicial	AP
1	356	227	[11 11]	96	256	4026	10	1e-3	0.43
2	356	227	[11 11]	96	256	4026	10	1e-3	0.49
3	356	227	[11 11]	96	256	4026	10	1e-3	0.54
4	650	300	[11 11]	96	256	4026	10	1e-3	0.59
5	650	300	[11 11]	96	256	4026	10	1e-3	0.62
6	750	350	[11 11]	96	256	4026	10	1e-3	0.66
7	750	350	[11 11]	96	256	4026	10	1e-3	0.79

8	950	375	[11 11]	96	256	4026	10	1e-3	0.80
9	950	375	[9 9]	96	256	4026	20	1e-3	0.83
10	1035	400	[9 9]	96	256	4026	20	1e-3	0.87
11	1035	400	[9 9]	96	256	4026	20	1e-3	0.90

Una vez obtenidos los resultados esperados de los entrenamientos de cada uno de los objetos puestos en estudio, se procede a tomar las configuraciones que han tenido el mejor rendimiento a la hora de detectar y clasificar la contaminación superficial. Con el objetivo de tener un punto de inicio para el desarrollo del detector y clasificador final, así mismo se realizan las diferentes configuraciones pertinentes hasta llagar a obtener una red que arroje los resultados deseados.

3.8.3.8.Desarrollo del Detector final

A continuación, se muestra la realización de la integración total en un solo detector que sea capaz de detectar los 3 contaminantes propuestos anteriormente, es decir, botellas plásticas, envases metálicos y totora, aprovechando lo aprendido durante el desarrollo de dichos entrenamientos realizados.

Tabla 11. Valores de entrenamiento de la CNN final

Fuente: (Autor)

Configuraciones de entrenamiento					
Configuración	Cantidad de Imágenes	Tamaño de Imágenes	Tamaño del filtro	Filtros primera capa	Taza Inicial de Aprendizaje
1	632 450	250x250x3 350x350x3	[11 11]	96	0.001

El script final tiene como nombre *FRCNNcontaminacionlaguna.m.* es aquel que contendrá todo el código para el entrenamiento del detector final, este a la vez será modificado según la estructura mostrada en la tabla 11.

Se procede a cargar el conjunto de datos de 1082 imágenes, de la cuales 632 son redimensionadas a 250x250 pixeles, esto se debe a que las imágenes adquiridas con la cámara del celular tienen mayor resolución y por otra parte las 450 imágenes redimensionadas a 350x350 pixeles.

Se procede a cargar la red pre-entrenada AlexNet, a la cual se le realizaron los siguientes cambios: En su primera capa convolucional conserva 96 filtros y de tamaño [9 9], los filtros de tamaño pequeño en operaciones convolutivas en las primeras capas permite que la realización de la operación sea más fina, contribuyendo a la conservación de la información al igual que lo hace el tamaño de la imagen en esta capa. La estructura es sencilla al igual que en la mayoría de las pruebas realizadas con los objetos de manera individual.

Con la CNN y las opciones de entrenamiento ya configuradas, se da inicio al entrenamiento de la red, para ello utilizamos el comando *'trainFasterRCNNObjectDetector'*. Como se ha comentado, es un entrenamiento lento. En una primera prueba, con el etiquetado de tres categorías contaminantes y teniendo 1082 imágenes en un ordenador portátil tardaba alrededor de 8 horas.

La tabla12, muestra el desarrollo y estadísticas del entrenamiento del detector final.

Tabla 12. Datos de entrenamiento detector final

Fuente: (Autor)

Command Window

```
*************************************************************************
Training a Faster R-CNN Object Detector for the following object classes:

* botellaplastica
* envasemetalico
* totora

Step 1 of 4: Training a Region Proposal Network (RPN).
Training on single CPU.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     RMSE     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:01 |       37.50% |         1.08 |          0.0010 |
|       1 |          50 |       00:00:20 |       94.49% |         0.67 |          0.0010 |
|       1 |         100 |       00:00:36 |      100.00% |         1.10 |          0.0010 |
|       1 |         150 |       00:00:53 |      100.00% |         0.94 |          0.0010 |
|       1 |         200 |       00:01:09 |      100.00% |         0.89 |          0.0010 |
|       1 |         250 |       00:01:25 |       98.44% |         1.66 |          0.0010 |
|       1 |         300 |       00:01:40 |       98.44% |         1.51 |          0.0010 |
|       1 |         350 |       00:01:57 |       97.66% |         0.67 |          0.0010 |
|       1 |         400 |       00:02:14 |       98.44% |         1.88 |          0.0010 |
|       1 |         450 |       00:02:29 |      100.00% |         0.60 |          0.0010 |
|       1 |         500 |       00:02:46 |       98.44% |         0.82 |          0.0010 |
fx >>
```

Luego de finalizar el entrenamiento la red está lista para ser evaluada sobre un conjunto de imágenes (*testData*), para ello se utiliza la métrica de precisión promedio, idealmente, la precisión sería 1 en todos los niveles de recuperación, en nuestro primer detector final, la precisión promedio es ~ 0.65. tal como se puede apreciar en la figura 56.

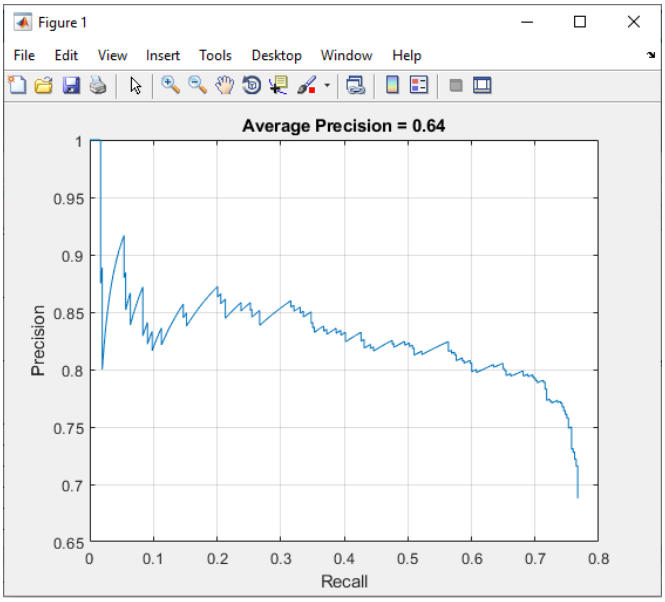


Figura 52. Curva de precisión/recuperación del detector final

Fuente: (Autor)

Se observa un aumento considerable de la cantidad de iteraciones de las épocas esto se da a causa del nuevo valor total de las imágenes obtenidas, que corresponden a 1250 imágenes con 3210 etiquetas, provenientes de las bases de datos de las imágenes de los objetos entrenados individualmente.

Algunas imágenes de prueba reflejan el comportamiento del detector final creado, las cuales se muestran a continuación en las figuras 57, 58 y 59. El programa utiliza el color amarillo para identificar botellas plásticas, el azul para identificar envases metálicos y el rojo para identificar totora.

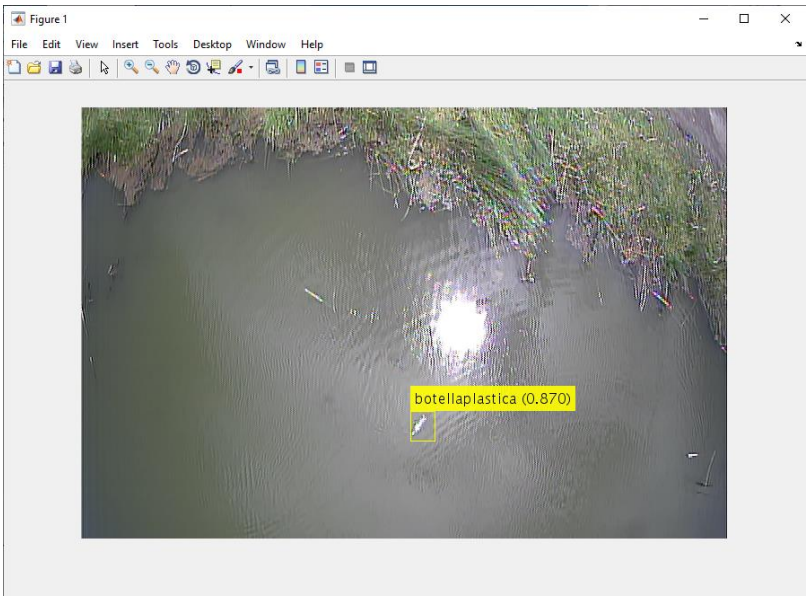


Figura 53. Detección de botella plástica

Fuente: (Autor)

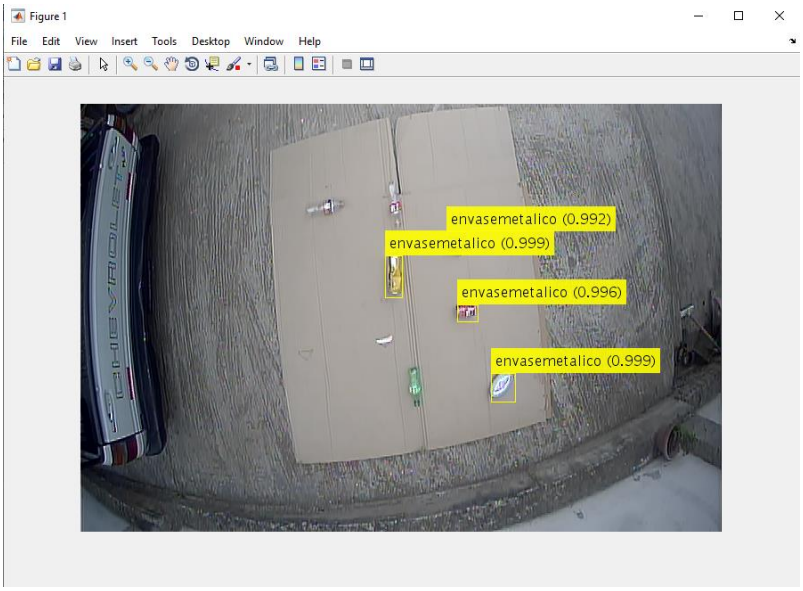


Figura 54. Detección de envase metálico

Fuente: (Autor)

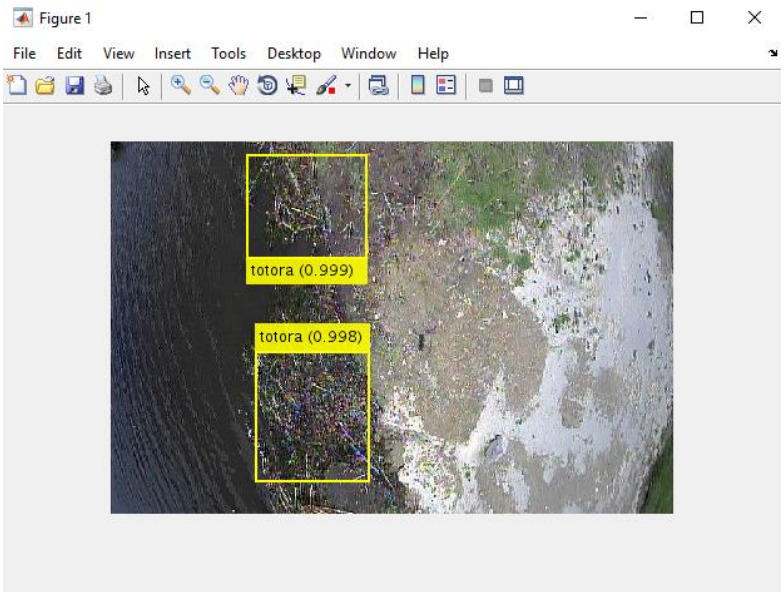


Figura 55. Detección de totora

Fuente: (Autor)

El resultado obtenido tras el primer entrenamiento de la red final no cumple con lo esperado, por tal motivo surge la necesidad de realizar las diferentes modificaciones de la red hasta obtener una red que arroje los resultados deseados.

Tabla 13. Precisión promedio del detector final

Fuente: (Autor)

Configuración	Numero de Imágenes	Capa de Entrada - Dimensiones	Tamaño del Filtro	Filtros primera capa	Filtros segunda capa	FC	Épocas	Taza de aprendizaje Inicial	AP
1	1080	227	[11 11]	96	256	4026	10	1e-3	0.70
2	1355	227	[11 11]	96	256	4026	10	1e-3	0.73
3	1400	227	[11 11]	96	256	4026	10	1e-3	0.75
4	1550	300	[11 11]	96	256	4026	10	1e-3	0.79
5	1700	300	[11 11]	96	256	4026	10	1e-3	0.82
6	1876	350	[9 9]	96	256	4026	10	1e-3	0.85
7	1924	350	[9 9]	96	256	4026	10	1e-3	0.87
8	2104	375	[9 9]	96	256	4026	10	1e-3	0.89
9	2381	400	[9 9]	96	256	4026	20	1e-3	0.90
10	2523	400	[9 9]	96	256	4026	20	1e-3	0.91

3.8.4. IMPLEMENTACIÓN DEL SISTEMA DE ADQUISICIÓN DE DATOS

En la presente sección se detalla el proceso llevado a cabo para la implementación del sistema, el cual se muestra en la figura 60, en el que se describe y consta de: microcontrolador, módulo de sensores, módulo GPS, módulo de video, seguido a esto se analiza el módulo de comunicación inalámbrica y finalmente el servidor.

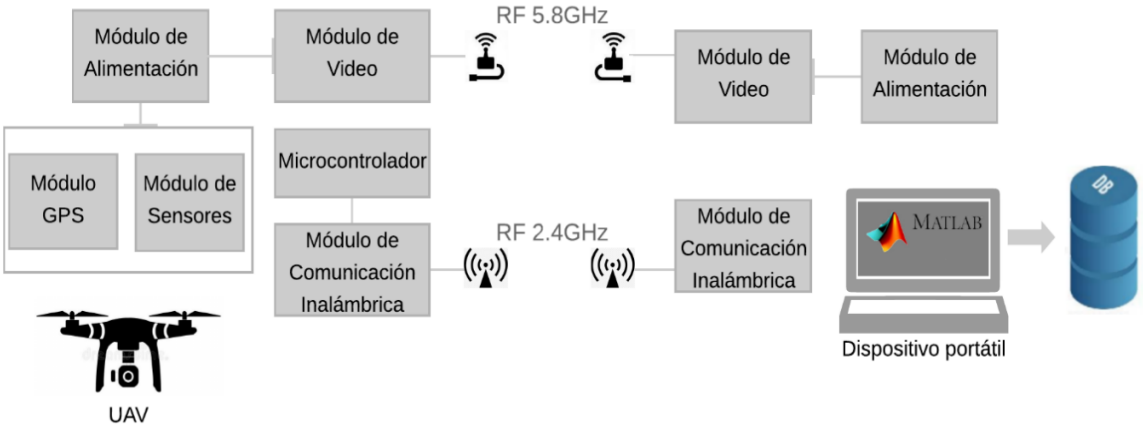


Figura 56. Sistema de adquisición de datos

Fuente: (Autor)

A continuación, en la figura 61 muestra el prototipo a implementar con cada uno de los módulos utilizados para llevar a cabo la recepción de datos en el ordenador para su interpretación correspondiente.

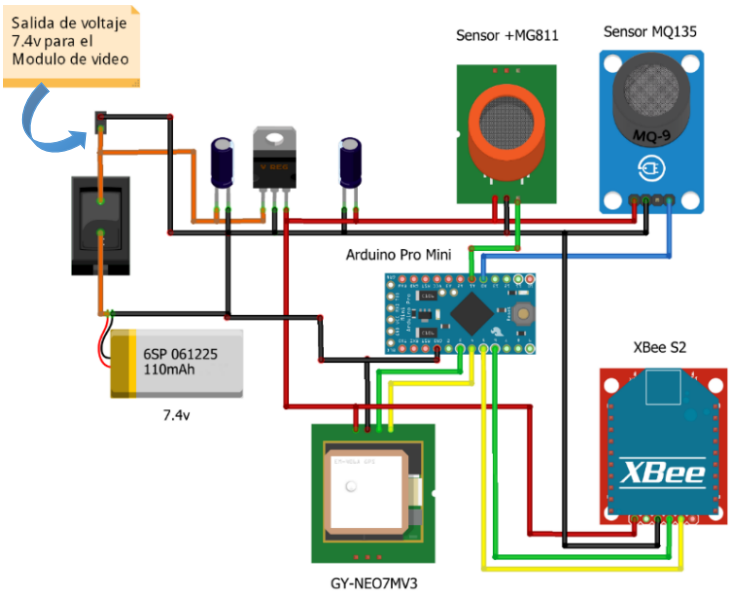


Figura 57. Módulo de sensores

Fuente: (Autor)

El sistema basa su funcionamiento en el uso del Arduino Pro Mini, dispositivo capaz de realizar la lectura y escritura de datos, puesto que los datos serán enviados a través de una comunicación serial hacia el software Matlab para su interpretación.

Cabe mencionar que el sistema tiene por separado una salida de voltaje de 7.4v para la alimentación del módulo de video, el cual se describe más adelante.

3.8.4.1.Módulo de Sensores

Son elementos fundamentales para el desarrollo de este trabajo, puesto que miden las variables físicas presentes en el medio ambiente y se encuentran agrupados para formar la interfaz de comunicación I2C.

Para esta aplicación, los sensores utilizados son los que se describen a continuación:

Tabla 14. Sensores utilizados

Fuente: (Autor)

Sensores	Variables	Pines
+MG811	Dióxido de Carbono (CO2)	Vcc, Gnd, D0, A0
MQ-135	Metano (CH4)	Vcc, Gnd, D0, A0

3.8.4.2.Módulo GPS

EL módulo GPS es uno de los principales elementos para el desarrollo del presente trabajo, debido a que, es el elemento que permite conocer la ubicación exacta del Drone mediante las coordenadas Latitud y Longitud.

El módulo utilizado es el NEO-7M-000, el mismo que resulta ser un dispositivo UART (*Universal Asynchronous Receiver-Transmitter*), además incluye una antena cerámica de alta ganancia, permitiéndole así, mejorar la receptividad de la señal.

3.8.4.3.Microcontrolador

Se ha utilizado la tarjeta de desarrollo ARDUINO Pro Mini (PataganiaTec, 2016), el cual cumple con la función de almacenar los datos arrojados por los sensores antes mencionados para su posterior conversión, así mismo adquiere las coordenadas mediante el GPS, además puede establecer cuando debe el módulo de comunicación inalámbrica iniciar la comunicación con otro modulo para enviar los datos.

Para realizar la transmisión de los datos arrojados por cada uno de los módulos se ha optado por realizar un algoritmo de adquisición de datos, en el que se puede apreciar en la figura 74.

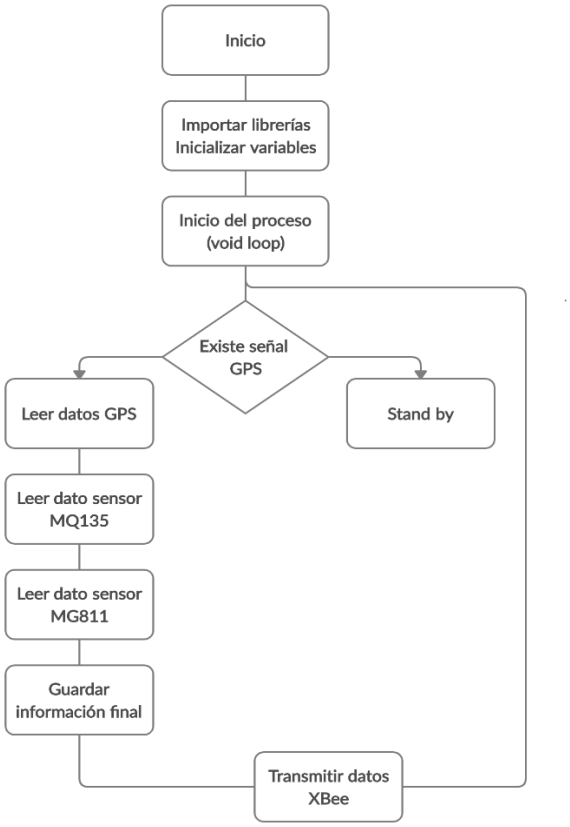


Figura 58. Diagrama de flujo para la transmisión de datos

Fuente: Autor

Una vez que el microcontrolador recibe los datos de los módulos, este envía la información cada 2 segundos utilizando una trama de 4 campos separados por una coma al módulo de recepción que estará conectado al ordenador. En la siguiente tabla 15, muestra la trama de información, con su estructura y formato.

Tabla 15. Trama de Información

Fuente: (Autor)

Latitud	Longitud	Dióxido de Carbono	Metano
+/-: Norte/Sur deg: grados d: decimales	+/-: Este/Oeste deg: grados d: decimales	ValorMG811: CO2 ppm: partes por millón	ValorMQ135: CH4 ppm: partes por millón

3.8.4.4.Módulo de comunicación inalámbrica

En la parte final del sistema, se procede a explicar el funcionamiento de los módulos de comunicación inalámbrica, tales como, las configuraciones iniciales y la forma de transmisión de los datos.

Para el proceso de la comunicación se ha empleado la tecnología ZigBee y se han elegido los módulos XBee S2C (ElectronicWings, 2018), quienes serán los encargados de enviar y recibir la trama de datos. Para empezar a trabajar con los módulos XBee S2C, es necesario configurarlos a nuestro interés mediante el programa XCTU, para realizar aquello es necesario seguir los siguientes pasos:

1. Se procede a conectar el módulo a la computadora, el será reconocido como un "Puerto serie USB". Tenemos que conocer el número COM dado a este dispositivo para especificarlo en la X-CTU (en nuestra prueba, COM6 fue el valor dado por Windows). Además, la velocidad de transmisión predeterminada para los módulos XBee es de 9600.

2. Luego aparece la pantalla principal del modo de trabajo de configuración. Ahí se ingresa cada uno de los parámetros que servirán para trabajar, uno en modo de Coordinador y el otro en modo de Router.
   Los módulos se deben configurar de la siguiente manera tal como muestra la tabla 16.

Tabla 16. Valores de configuración

Fuente: (Autor)

Indicador	Coordinador	Router
ID PAN ID	2018	2018
JV Channel Verification	Disabled	Enabled
CE Coordinator Enable	Enabled	Disabled
DL Destination Address Low	419B5B72	419B5B99
DH Destination Address High	13A200	13A200

3. Por último, se escribe las configuraciones de los módulos mediante *"Write"*.

Una vez configurados los módulos a nuestro interés se puede ya empezar la comunicación entre el sistema de sensores y el ordenador, el Router será instalado en el circuito del sistema de sensores, mientras que, el Coordinador de la red en el ordenador será quien reciba la información.

3.8.4.5.Módulo de Video

El sistema de transmisión de FPV como muestra la figura 59, utiliza los siguientes dispositivos: transmisor TS832, receptor RC832, MyGica Capit (Captura de video USB), batería de 7.4v a 1000mAh para alimentar el transmisor, batería de 7.4v a 5200mAh para alimentar el rector y una cámara fpv. Los módulos de video mencionados operan en la banda de 5.8 GHz, transmiten una calidad de video de 1080 x 720 a través de una cámara fpv encargada de capturar la imagen del entorno de la Laguna que será procesada mediante los algoritmos de inteligencia artificial para detectar y clasificar contaminación superficial con la ayuda del software Matlab.

Figura 59. Sistema de transmisión FPV

Fuente: (Autor)

Los módulos del sistema de video deben estar concertados a las antenas omnidireccionales para ser polarizadas al momento de conectar a la fuente de alimentación.

Para iniciar la comunicación entre los dispositivos TX y RX, se debe elegir la frecuencia y el canal de comunicación en cada uno de los dispositivos, se cambia fácilmente los canales y las bandas de frecuencia presionando los botones, una vez configurados los módulos empieza a realizar la comunicación, en la computadora es necesario instalar el driver del dispositivo MyGica para que inicie la captura de video por USB.

3.8.5. INTERFAZ GRÁFICA

Contando con la CNN reentrenada y con el sistema de adquisición de datos de los diferentes módulos funcionando, se ha optado por la creación de la interfaz gráfica de usuario con la ayuda de Matlab, capaz de reconocer y clasificar los contaminantes superficiales en la Laguna de Colta. Así mismo se realizará un interfaz de usuario para visualizar los datos geo referenciales de la concentración de contaminación superficial, esto se llevará a cabo con el ayuda del motor gráfico 3D de Unity.

3.8.5.1.Matlab

La figura 64 muestra la pantalla principal de la interfaz de usuario, la cual posee imágenes de fondo y textos, para luego estar a la espera de presionar el botón de inicio correspondiente a la ventana del detector y clasificador.

Figura 60. Pantalla principal

Fuente: (Autor)

Figura 61. Interfaz gráfica de usuario principal

Fuente: (Autor)

En la figura 66 se presenta la interfaz gráfica para la adquisición de los frames de video, procesarlos con la CNN, la cual permitirá reconocer y clasificar los diferentes contaminantes superficiales, además, muestra los niveles de contaminación de forma proporcional. La

ventana consta con los siguientes botones: Activar GIS Laguna, Activar Cámara, Detectar y Clasificar, Stop, Adquisición Datos, Exportar Datos Json, Sensores y Regresar.

También, se puede visualizar 3 recuadros descritos a continuación: en el recuadro izquierdo se proyecta la imagen del mapa, en el del centro se visualiza el video en tiempo real y en el derecho se presenta la tabla de coordenadas de los objetos contaminantes detectados.

Además, se observa las coordenadas GPS del drone, el número de objetos detectados y la concertación de contaminación.

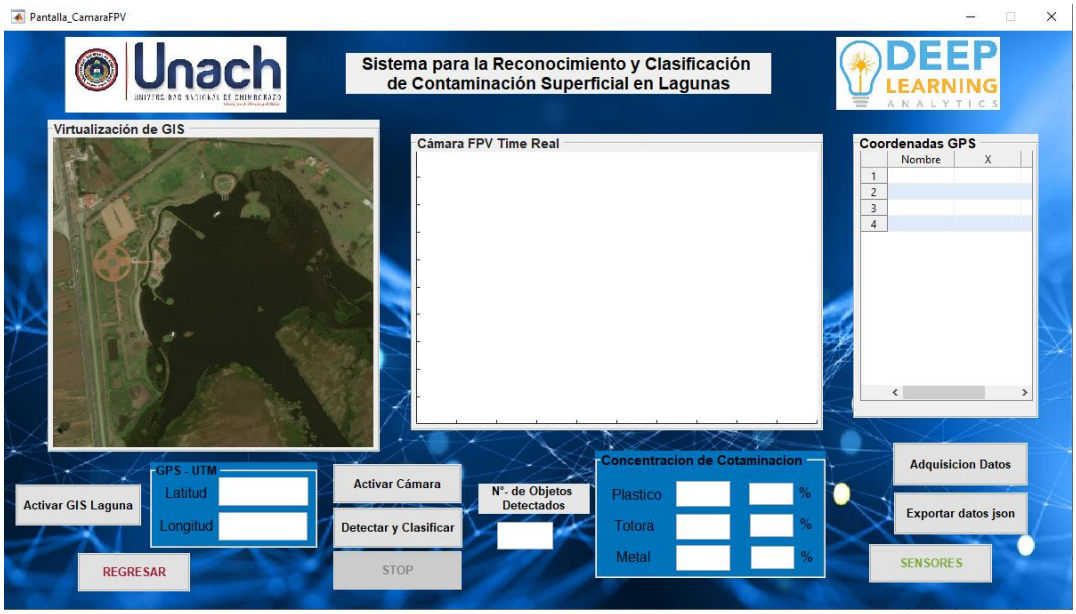


Figura 62. Pantalla de Detección y Clasificación

Fuente: (Autor)

Se indica en la figura 67 el proceso que realiza el botón Activar GIS Laguna, el mismo que permite cargar la imagen de la Laguna y así poder observar la posición del drone en tiempo real.

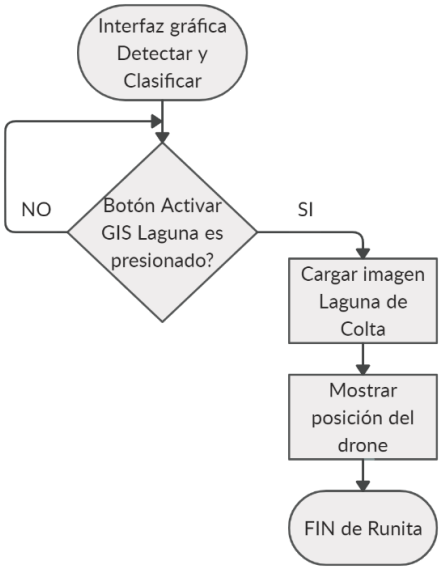


Figura 63. Diagrama de flujo del botón Activar GIS Laguna

Fuente: (Autor)

En la figura 68 se observa el diagrama de fuljo del botón Activar Cámara, el cual permite visualizar la imagen de video en el recuadro del centro capturada por la cámara fpv.

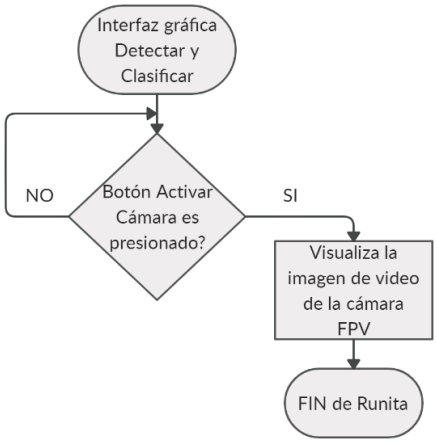


Figura 64. Diagrama de flujo del botón Activar Cámara

Fuente: (Autor)

Se detalla en la figura 69 el funcionamiento del botón Detectar y Clasificar, el cual inicia la adquisición de los frames de video, estos a la vez serán procesados por la CNN reentrenada y realizara las predicciones correspondientes.

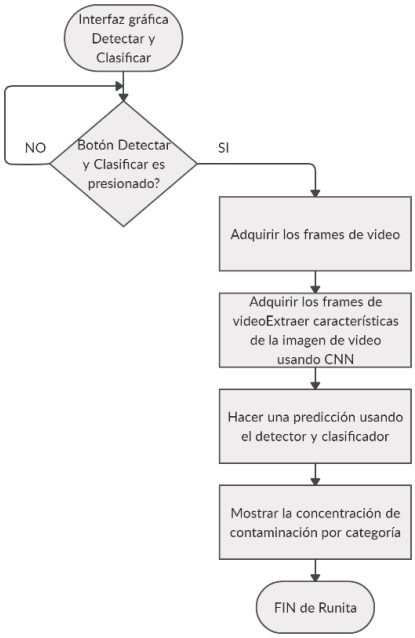


Figura 65. Diagrama de flujo del botón Detectar y Clasificar

Fuente: (Autor)

En la figura 70 se da a conocer el proceso que ejecuta el botón Adquisición de Datos, el mismo que habilita la adquisición de las etiquetas de los objetos detectados con sus respectivas coordenadas.

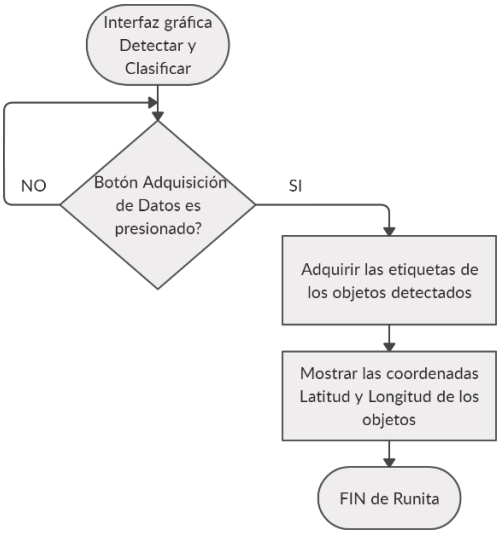


Figura 66. Diagrama de flujo del botón Adquisición de Datos

Fuente: (Autor)

Se describe en la figura 71 el diagrama de flujo del botón Exportar Datos Json, el cual permite convertir los datos a formato Json y crear un archivo (*.txt*) de coordenadas de los objetos contaminantes detectados.

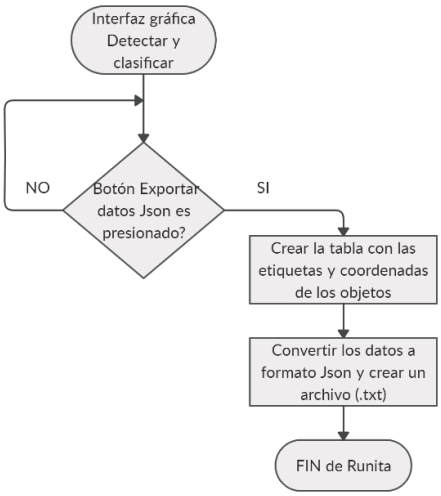


Figura 67. Diagrama de flujo del botón Exportar Datos Json

Fuente: (Autor)

Se detalla en la figura 72 el diagrama de flujo del botón Sensores, el mismo que al ser presionado abre la interfaz gráfica de usuario para la adquisición de los niveles de contaminación proporcionados por los sensores ambientales.

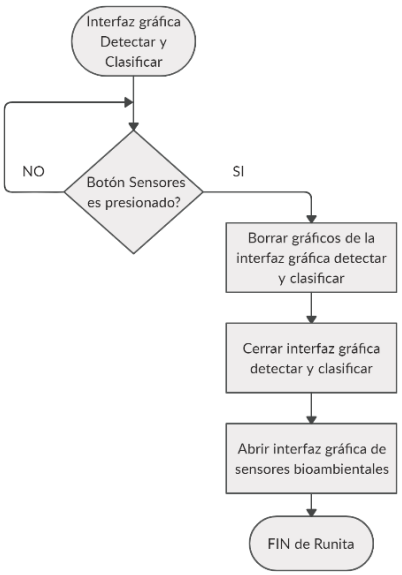


Figura 68. Diagrama de flujo del botón Sensores

Fuente: (Autor)

A continuación, en la figura 73 se da conocer el funcionamiento del botón Regresar, el cual borra los gráficos, cierra el interfaz de usuario de detección y clasificación y abre el interfaz de usuario de inicio.

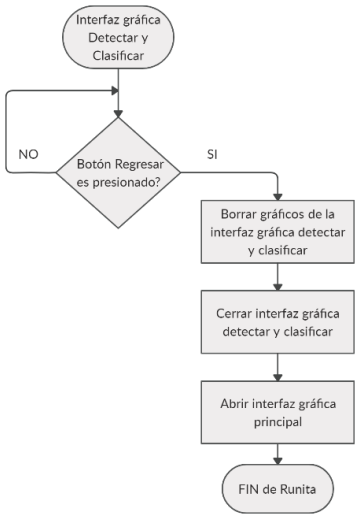


Figura 69. Diagrama de flujo del botón Regresar

Fuente: (Autor)

La interfaz gráfica para obtener los datos de los sensores de dióxido de carbono y metano se observa en la figura 74, en la cual constan los botones: Activar GIS Laguna, Activar Sensores, Adquisición Datos Sensores, Exportar Datos Json, Regresar y Salir.

Así mismo, se observa 3 recuadros descritos de la siguiente manera: en el recuadro izquierdo se arroja la imagen del mapa de la Laguna de Colta, en el recuadro del centro se visualizará los datos de los sensores en forma gráfica y en el recuadro derecho mostrará las coordenadas de los sensores.

Además, la interfaz gráfica muestra las coordenadas de posición del drone y la concentración de gas contaminante.

Figura 70. Pantalla de Sensores Ambientales

Fuente: (Autor)

Se indica en la figura 75 el diagrama de flujo del botón Activar GIS, en mismo que permite cargar una imagen de la Laguna de Colta en el recuadro superior izquierdo.

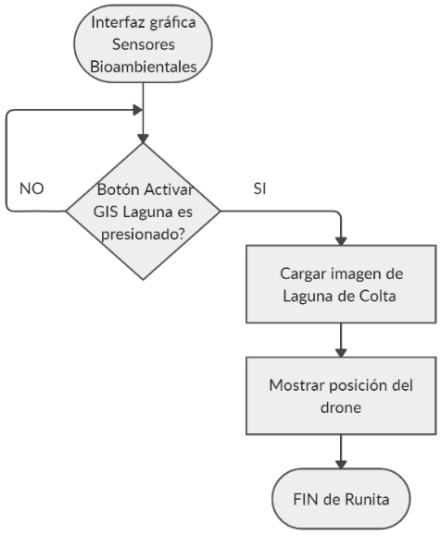


Figura 71. Rutina botón Activar GIS Laguna

Fuente: (Autor)

En la figura 76 se describe la rutina del botón Activar Sensores, el cual permite el inicio del proceso de adquisición de los niveles de dióxido de carbono y metano.

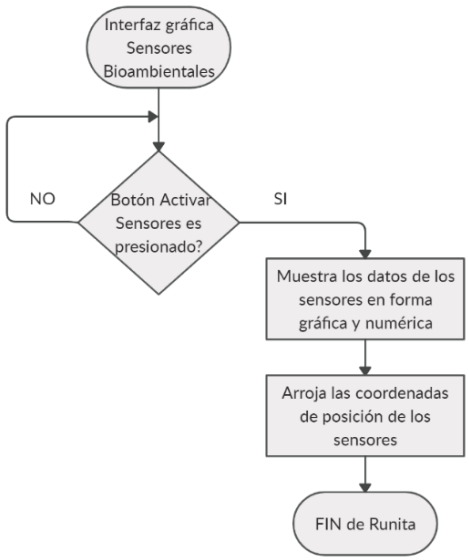


Figura 72. Rutina botón Activar Sensores

Fuente: (Autor)

En la figura 77 se describe el diagrama de flujo del botón Adquisición de Datos, el cual habilita la adquisición de los datos de los sensores en la tabla.

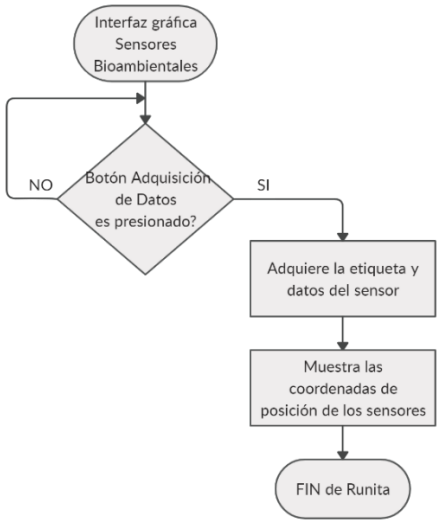


Figura 73. Rutina botón Adquisición de Datos

Fuente: (Autor)

En la figura 78 se describe el proceso del botón Exportar Datos, el cual genera el archivo (.txt) de los gases contaminantes en formato Json, además, obtiene las coordenadas de los puntos de concentración de gas.

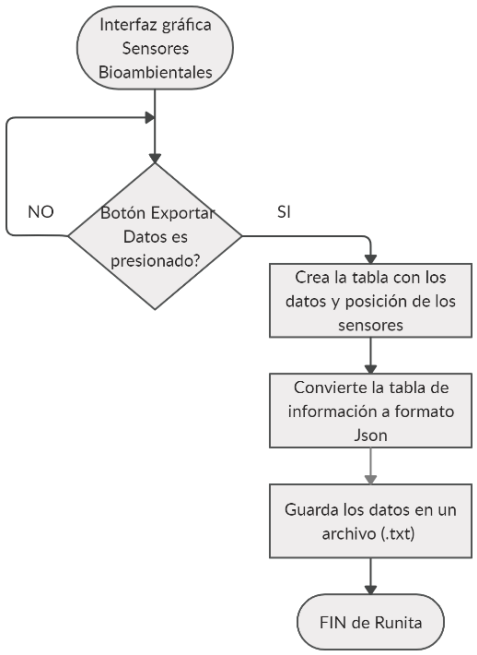


Figura 74. Rutina del botón Exportar Datos

Fuente: (Autor)

Se indica, en la figura 79 el diagrama de flujo del botón Regresar, el mismo que cierra la interfaz gráfica de los sensores ambientales y abre la interfaz gráfica del detector y clasificador.

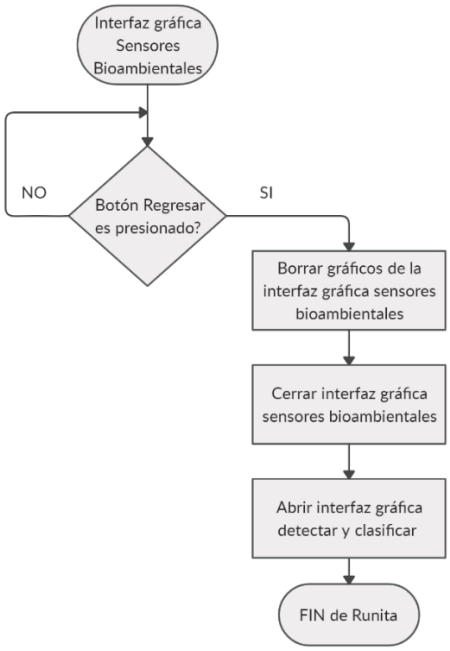


Figura 75. Rutina botón Regresar

Fuente: (Autor)

3.8.5.2.Unity

Para desarrollar la interfaz de usuario (UI: User Interface) como ya se mencionó se hace uso del motor gráfico 3D de Unity. La figura 80, muestra el diagrama de bloques para el desarrollo de la UI en Unity.

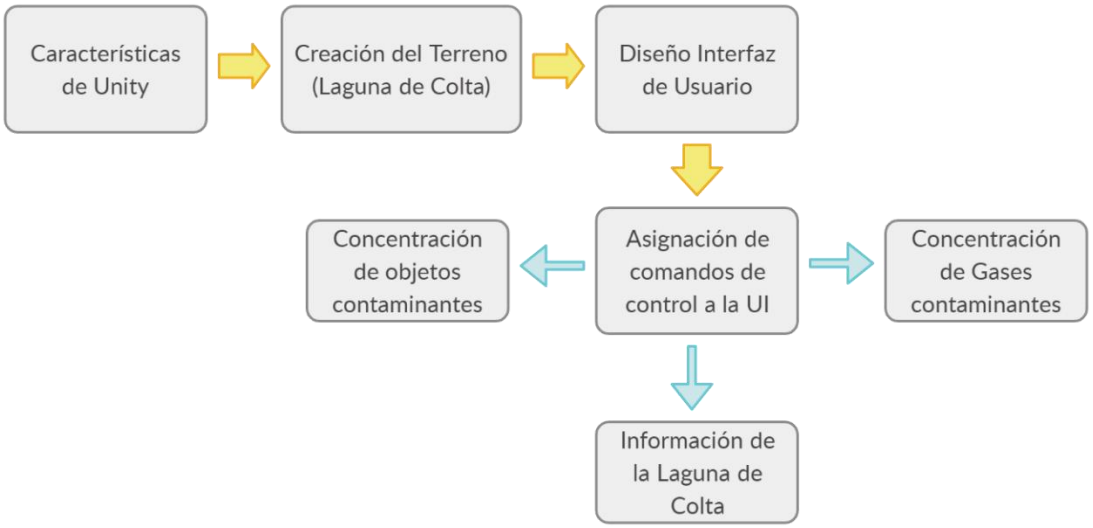


Figura 76. Diagrama de bloques para el desarrollo de la UI en Unity

Fuente: (Autor)

Todas las acciones se realizan desde la interfaz de usuario que será totalmente manual, y se dividirá en 3 interfaces como se describen a continuación:

- **UI Logo. -** La primera interfaz en mostrarse, como muestra la figura 81, tiene como único propósito mostrar el logo del creador de la aplicación, el cual tendrá 1 botón principal: START.

Figura 77. UI Logo

Fuente: (Autor)

- **UI Portada. -** La UI portada de la aplicación como muestra la siguiente figura 82, tendrá 2 botones principales: RETURN y START VIRTUALIZATION. Esta interfaz se encargará de mostrar una imagen perspectiva de la realidad aumentada.

Figura 78. UI Portada

Fuente: (Autor)

- **UI Menú principal. -** El menú principal de la aplicación tal como se muestra en la figura 83, tendrá 3 botones principales: salir de la UI menú principal, mostrar panel de objetos contaminantes y mostrar panel de gases contaminantes.

El propósito de la UI Menú principal, será mostrar un panel de ejecución ubicado a la derecha quién mostrará la información referencial de la Laguna de Colta, concentración de objetos contaminantes y concentración de gases contaminantes, tal como se puede apreciar en las siguientes figuras 84 y 85, además, incluye un botón para generar su geolocalización y un botón para regresar al panel de información.

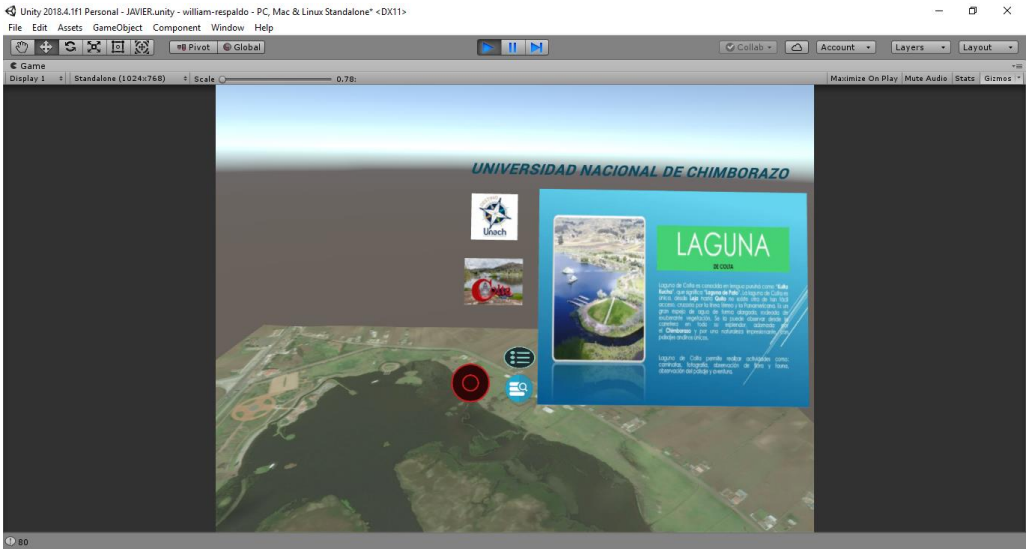


Figura 79. UI Menú principal

Fuente: (Autor)

Figura 80. Panel de concentración de objetos contaminantes

Fuente: (Autor)

Figura 81. Panel de concentración de gases contaminantes

Fuente: (Autor)

CAPITULO IV

4. RESULTADOS Y DISCUSIÓN

En este capítulo se muestran los resultados de las pruebas de detección y clasificación de la técnica de aprendizaje profundo (Faster R-CNN). Para la prueba del sistema de reconocimiento y clasificación se va a tomar un video con calidad de 640 x 480 pixeles, durante el día y en buenas condiciones climáticas, el mismo que va a tener un fondo en movimiento. Además, para evaluar el desempeño del método propuesto en la detección y clasificación de la contaminación superficial se dispone de 45 imágenes por clase que servirá para evaluar su confiabilidad por medio de la curva de precisión/recuperación (PR).

4.8. Sistema para la adquisición de datos de sensores y captura de imagen de video

Con la finalidad de obtener un correcto funcionamiento de los diferentes módulos que conforman el sistema de adquisición de datos y video, se ejecutaron varios experimentos en relación a cada una de las etapas.

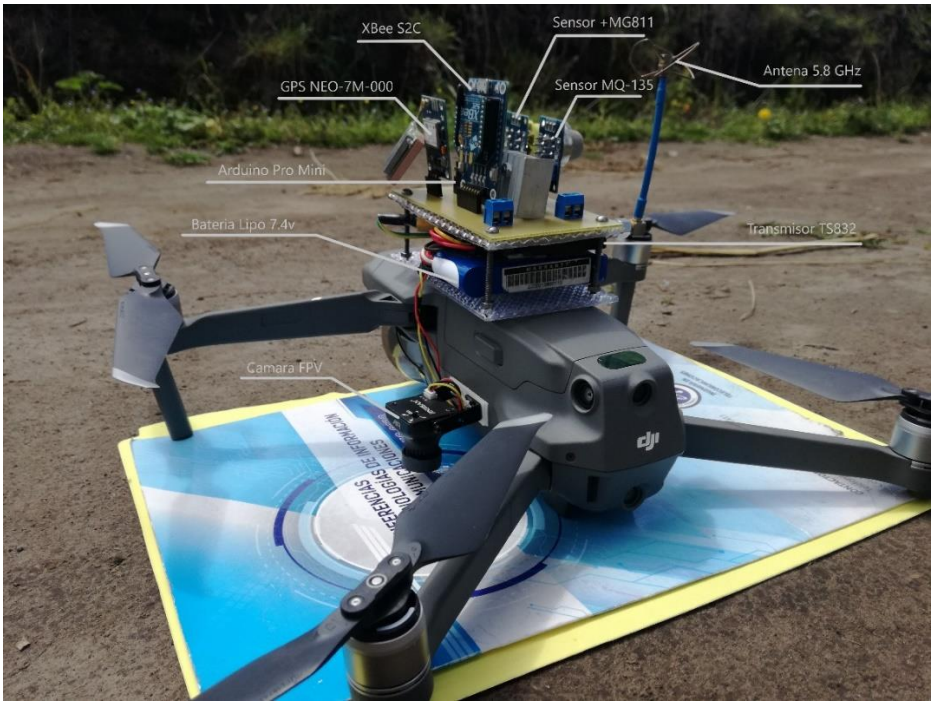


Figura 82. Sistema para la adquisición de datos

Fuente: (Autor)

Se parte con el análisis de los canales de comunicación de video en base a los valores de latencia arrojados por la banda y frecuencias de operación.

En una primera prueba en conocer el tiempo de enlace entre el modulo transmisor y el módulo de recepción tarda 63 mili segundos aproximadamente, a continuación, se aprecia en la tabla 17 la latencia con cada uno de las diferentes bandas y frecuencias de operación.

Tabla 17. Latencia Generada por cada banda de operación

Fuente: (Autor)

Banda de Operación	**Canal de Comunicación**	**Frecuencia (MHz)**	**Tiempo en video (s)**	**Latencia (ms)**
A	1	5865	10	63
	2	5845	12	58
	3	5825	12	55
	4	5805	10	63
	5	5785	13	59
	6	5765	15	63
	7	5745	10	66
	8	5725	11	70
B	1	5733	9	65
	2	5752	10	59
	3	5771	13	65
	4	5790	14	59
	5	5809	10	62
	6	5828	12	65
	7	5847	8	63
	8	5866	10	71
E	1	5705	8	80
	2	5685	10	85
	3	5665	12	79
	4	5665	11	75
	5	5885	15	65
	6	5905	13	60

	7	5905	12	63
	8	5905	10	65
F	1	5740	15	70
	2	5760	14	65
	3	5780	14	70
	4	5800	11	63
	5	5820	10	78
	6	5840	12	80
	7	5860	9	85
	8	5880	12	87
r	1	5658	12	60
	2	5695	10	70
	3	5732	14	65
	4	5769	13	75
	5	5806	14	83
	6	5843	12	85
	7	5880	10	92
	8	5917	13	89

4.9. Prueba del detector y clasificador en los *frames* de video

Para realizar las pruebas en los *frames* de video, buscamos puntos específicos como la accesibilidad y facilidad que posee el atractivo (Laguna de Colta) para movilizar el drone, mismo que se pudo plasmar en un video las orillas de la superficie de la Laguna.

Seleccionamos una secuencia de video de la orilla y *frame* a *frame* realizará el procesamiento. Al momento de ejecutar la red para que analice los *frames* de video se obtuvo resultados satisfactorios, como se observa en las siguientes figuras.

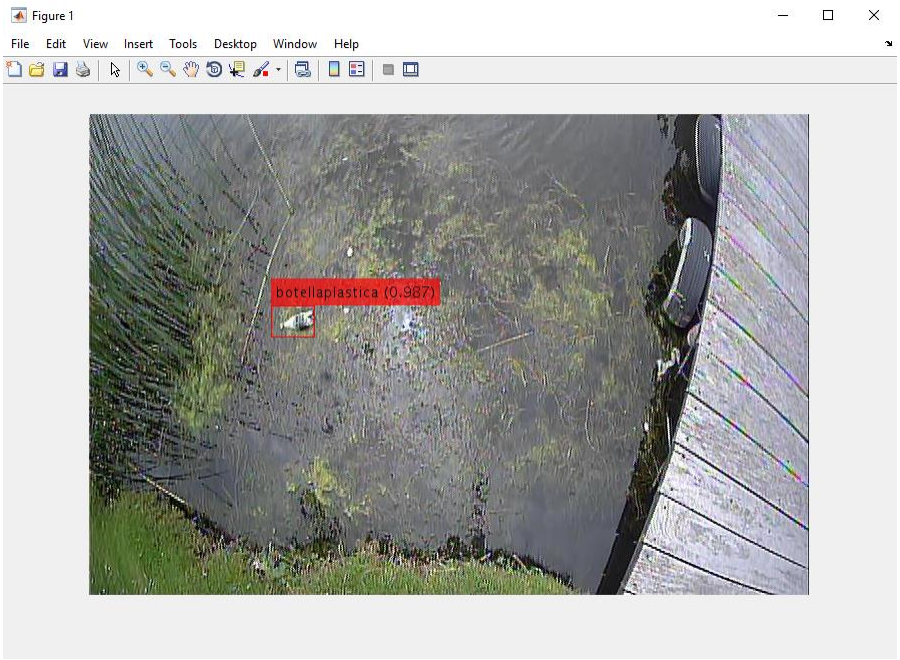


Figura 83. Detección de Botella plástica

Fuente: (Autor)

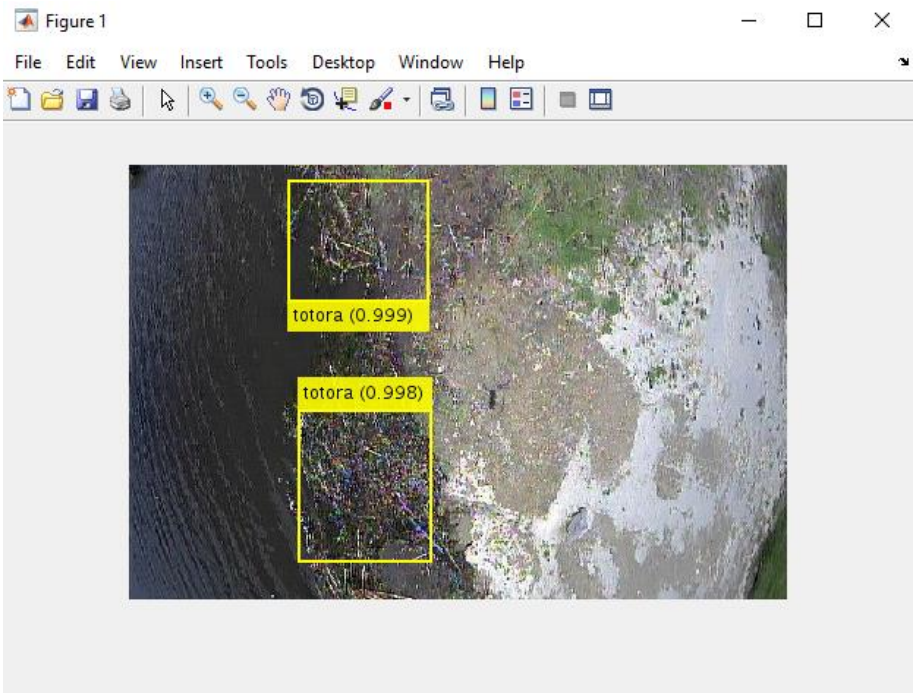


Figura 84. Detección de Totora

Fuente: (Autor)

Cabe mencionar que para realizar las pruebas pertinentes a la detección y clasificación de la contaminación superficial en la Laguna de Colta se adaptaron los siguientes parámetros de plan de vuelo del drone, tal como se muestra en la tabla 20.

Tabla 18. Parámetros de plan de vuelo

Fuente: (Autor)

Característica	Valor
Altitud del drone	4 m
Resolución de la cámara fpv	640x480 pixeles
Velocidad del drone	0.8 m/s
Puntos de referencia	17 puntos

En la figura 85, se muestra una de las trayectorias generadas con mayor eficiencia al momento de empezar a ejecutar el sistema de detección y clasificación.

Figura 85. Trayectoria Generada

Fuente: (Autor)

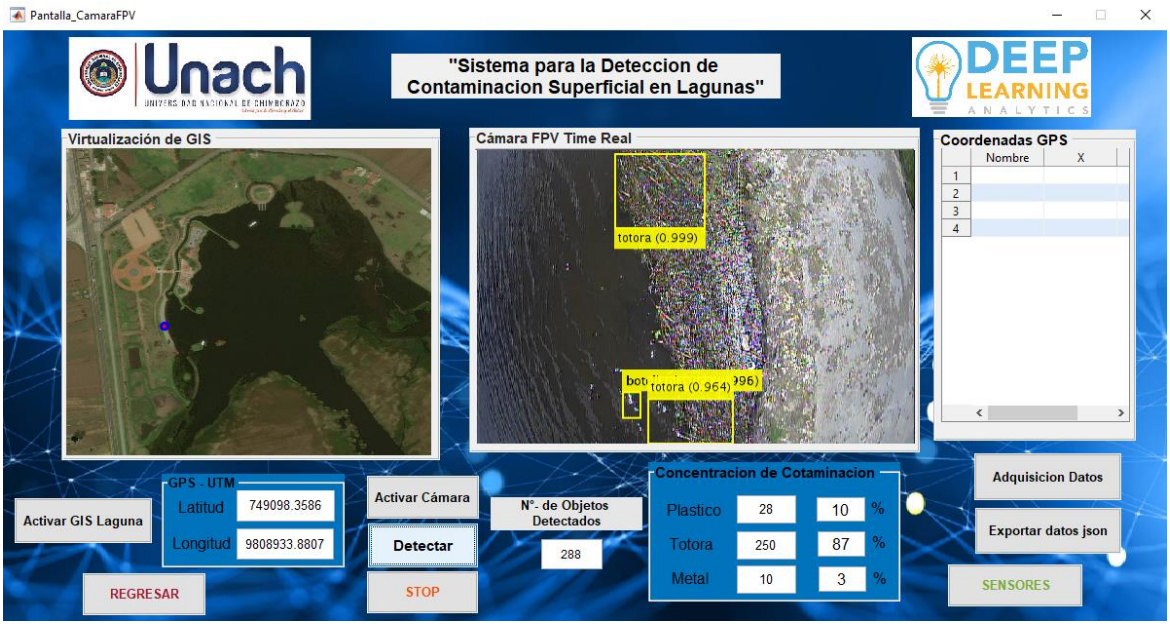


Figura 86. Resultado de la detección de totora

Fuente: (Autor)

Como se puede observar, detecta todos los objetos contaminantes correctamente, sin confundir ningún objeto con otro y detectando todos los que aparecen en ese *frame*. Además, a ello se puede observar la posición del drone en tiempo real, que a su vez el recorrido es referenciado por una línea de color azul.

Por último, se añadió un contador de etiquetas, es decir, a lo largo del proceso de cada frame, se pudiese ir contando las veces que se detectaba un objeto correspondiente a cada frame, para así llegar a tener un control de la concentración de contaminación superficial.

Los niveles de concentración de los objetos contaminantes en la superficie de la Laguna se describen a continuación: botellas plásticas (azul), envases metálicos (rojo) y totora (amarillo). La concentración de la contaminante totora es superior en comparación con los demás objetos.

En la figura 91, se puede observar los puntos georreferénciales de los objetos detectados sobre la superficie de la Laguna.

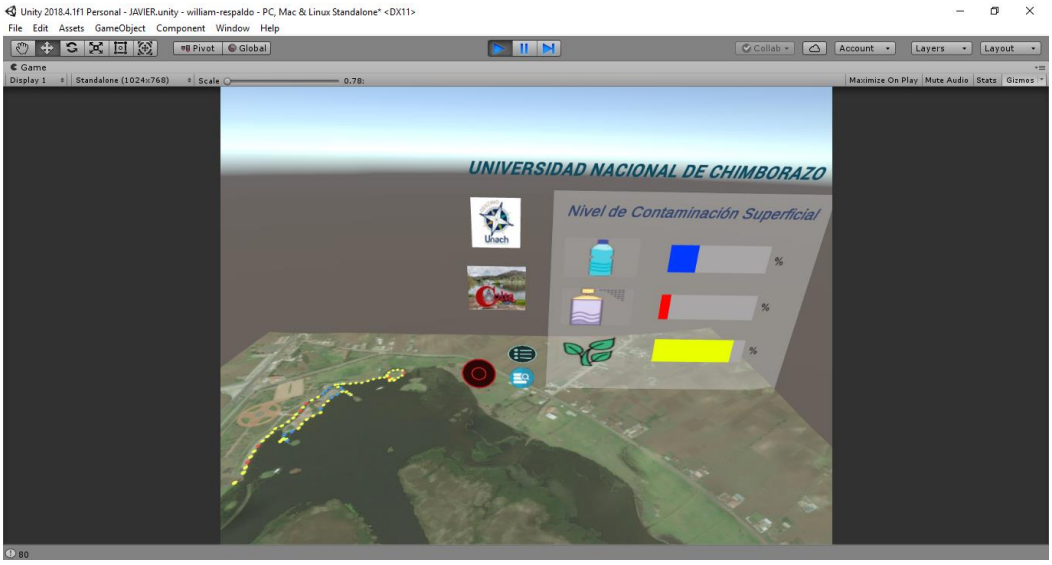


Figura 87. Geolocalización de los objetos contaminantes

Fuente: (Autor)

4.10. Resultados de Calidad del aire

En la figura 88, se observa los niveles de calidad del aire adquiridos de entre los 25 puntos referenciales sobre la Laguna de Colta, tras la trayectoria del drone con el módulo de sensores, donde se puede observar que los valores son equivalentes en cada uno de los puntos referenciales.

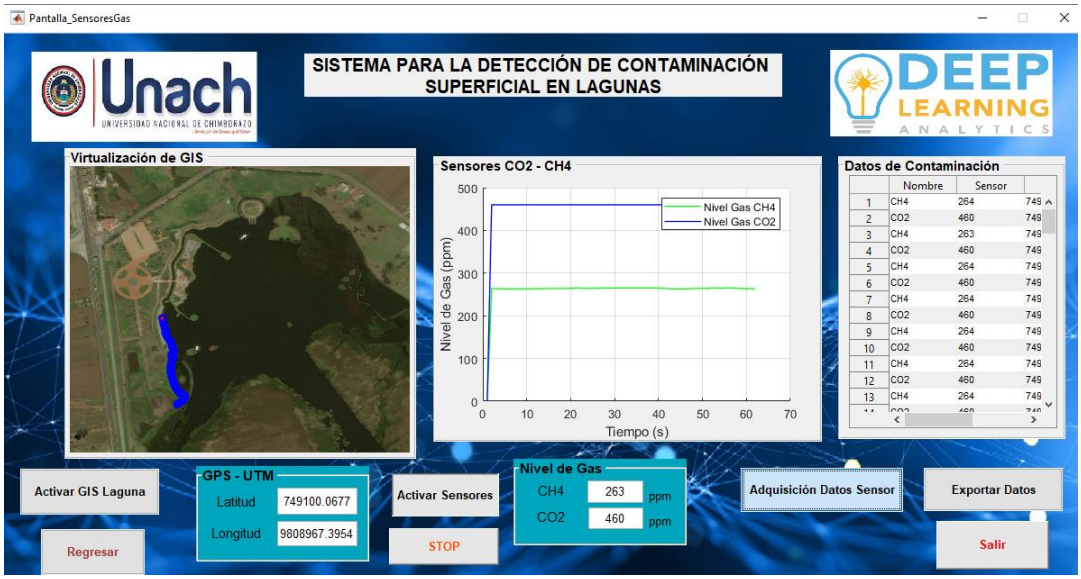


Figura 88. Resultado de calidad de aire

Fuente: (Autor)

Los niveles de calidad del aire adquiridos tienen diferente patrón, los mismos que se describen a continuación: sensor CO2 (azul) y el sensor CH4 (verde). La concentración del gas CO2 es superior en comparación al gas CH4.

Además, a ello se puede observar en la figura 89, los puntos georreferénciales de las muestras tomadas por cada uno de los sensores que miden la calidad del aire sobre la superficie de la Laguna.

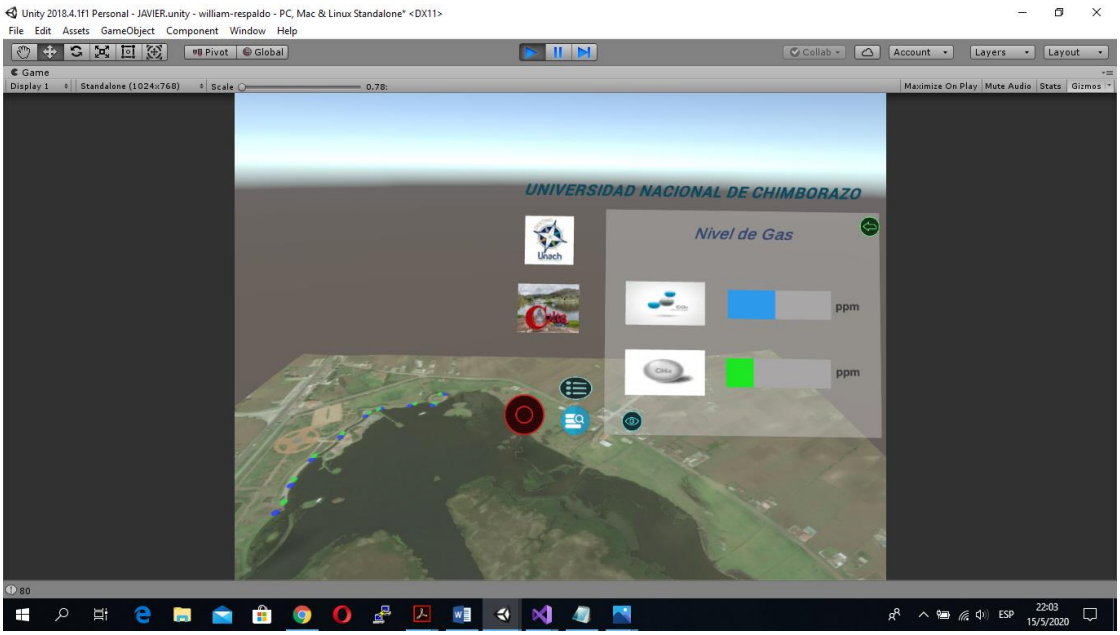


Figura 89. Geolocalización de los niveles de calidad del aire

Fuente: (Autor)

4.11. Evaluación de Precisión

Para evaluar el rendimiento del método propuesto en la detección y clasificación de contaminación superficial, se realizó una prueba de condición real en los conjuntos de datos proporcionados. A continuación, en la figura 94, se observa los resultados de la evaluación del método propuesto en términos de precisión.

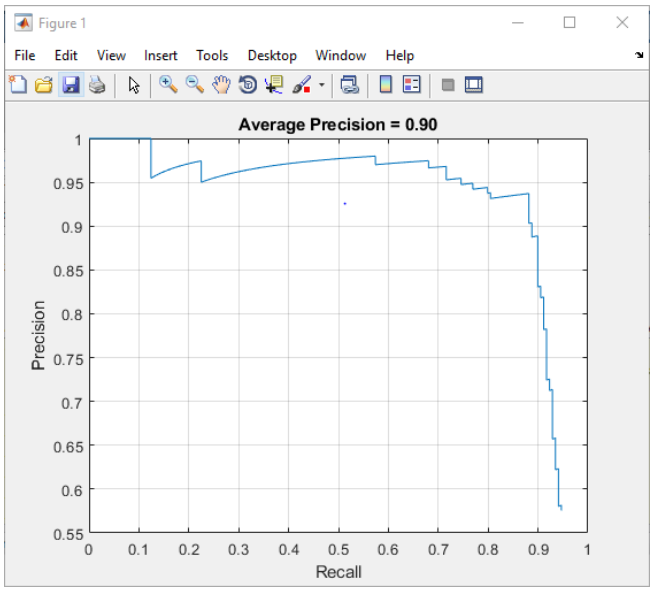


Figura 90. Curva de precisión del método propuesto

Fuente: (Autor)

Además de evaluar la eficiencia del método propuesto en términos de precisión, se calcula la tasa de clasificación errónea por medio del factor de sensibilidad. La sensibilidad se define como la fracción de datos relevantes (objetos detectados: botella plástica, envase metálico y totora) entre todos los resultados recuperados (todas las imágenes de objetos pertenecen al conjunto de datos).

Tabla 19. Resultados de evaluación de precisión del método

Fuente: (Autor)

Conjunto de datos	# de objetos contaminantes	TP	FN	Sensibilidad
Conjunto de datos " objetos contaminantes"	200	179	21	0.895
Otros conjuntos	280	248	32	0.886
Total	**480**	**350**	**130**	**0.89**

Según la gráfica de la tabla 19, la precisión del sistema es impresionante tanto en condiciones reales como en conjuntos de datos estándar.

La alta tasa de precisión en la aplicación del modelo propuesto a los datos reales conduce a un rendimiento confiable para utilizar el método propuesto en aplicaciones de visión artificial basadas en video.

4.12. Discusión

Como se puede observar de acuerdo con los valores de precisión arrojados por el método propuesto, el detector y clasificador que utiliza redes neuronales convolucionales es capaz de detectar y clasificar objetos contaminantes en condiciones casi en tiempo real con una precisión aceptable.

Sin embargo, esta se ve afectada debido a que los diferentes envases metálicos y botellas plásticos tienen características semejantes, es decir, que un objeto metálico puede ser detectado como un objeto plástico, debido a que la red aprende las características más relevantes como son los objetos plásticos, existiendo un error al momento de clasificar botellas plásticas y envases metálicos.

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.8. CONCLUSIONES

- Seguidas todas las pautas de este proyecto, y una vez completado el detector y clasificador basado en tecnología *Faster R-CNN* permiten llegar a una solución vigorosa, en cuanto a la detección y clasificación de los contaminantes presentes en la superficie de la Laguna de Colta se refiere y utilizando el software MATLAB.
- Estos resultados son bastante buenos, aunque en algunos *frames* de video se pudo apreciar que hay algunos objetos contaminantes que no llega a detectarlos, esto se debe a las distorsiones y ruidos presentes por un entorno no controlado que dificultan notablemente al momento de reconocer y clasificar.
- El proceso de visualización de cada uno de los frames de video es algo lento, por lo que no se podría ver en tiempo real, para que el procesamiento llegase a ver en tiempo real un *frame* no debería de tardar más de 1/30 de segundo.
- La cantidad de imágenes con objetos metálicos en la superficie de la Laguna es pequeña en relación con los objetos de botellas plásticas y totora, por lo que resulta difícil tener un gran número de imágenes que representen este contaminante, esto a su vez dificulta el entrenamiento del detector y clasificador.
- Se implementó un interfaz de usuario para una visualización oportuna, tanto de la información de los niveles de contaminación superficial, como también de los datos que arrojan los sensores bio-ambientales de concentración de gases contaminantes.
- El trabajo de investigación propuesto da un nuevo enfoque a los sistemas de detección y clasificación tradicionales, ya que al utilizar el aprendizaje profundo (Deep Learning) tiene una gran ventaja sobre las acciones de extracción de características y clasificación de imágenes, sabiendo que las CNN manejan grades cantidades de datos y son capaces de tomar decisiones con brevedad lo que las hace más robustas y confiables.

5.9. RECOMENDACIONES

- Para la captura de las imágenes de los objetos contaminantes superficiales con la cámara de video se debe tener en cuenta la cantidad de iluminación natural existente, ya que, la excesiva iluminación, produce el efecto de reflexión directa y esto puede provocar una pérdida de características del objeto. Por tal motivo se recomendable realizar la captura de la imagen cuando la iluminación natural no es excesiva.
- Si se requiere utilizar las redes neuronales especialmente las Redes Neuronales Convolucionales en Matlab, se recomienda utilizar las versiones actuales por delante de Matlab R2017a, visto que esto facilitaría el entrenamiento de la red y no generaría conflictos y errores.
- Al momento de trabajar con redes neuronales de gran tamaño y se utilizan base de datos de una cantidad de datos considerable, es recomendable disponer de un computador con buenas características, en cuanto a tarjeta de video y procesador, ya que estas presentan las limitaciones de vital importancia en el momento de entrenar o evaluar nuestra red.
- Al ejecutar el sistema de reconocimiento y clasificación de objetos contaminantes utilizando un drone, hay que tener en cuenta la altura a la cual se lo hace volar, ya que en considerables alturas se perdería algunas de las características principales y esto a su vez causar la confusión del objeto a detectar. En el caso de este proyecto se puede a una altura aproximada a 4 metros.
- El campo de inteligencia artificial, los *frameworks* han crecido a un ritmo acelerado durante los últimos años y es por ello que surjan nuevas librerías o versiones que mejoren el rendimiento, por tal motivo se recomienda actualizar el diseño regularmente.
- Entrenar por separado cada uno de las clases de objetos a detectar, llega a reforzar el entrenamiento de la red y facilita la detección de errores, así como también permite tener un inicio apto para el desarrollo del detector final.

BIBLIOGRAFÍA

Alex Krizhevsky, I. S. (2012). ImageNet Classification with Deep Convolutional. *papers*, 2-6.

Araujo, L. (16 de Abril de 2018). *Pooling Layer*. Obtenido de artificial-inteligence: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/pooling_layer.html

Burgal, J. U. (5 de Diciembre de 2018). *Deep Learning básico con Keras (Parte 4): ResNet*. Obtenido de https://enmilocalfunciona.io/deep-learning-basico-con-keras-parte-4-resnet/

Cajamarca, M. (6 de Junio de 2019). *Planeta CHABOT*. Obtenido de Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo: https://planetachatbot.com/inteligencia-artificial-aprendizaje-autom%C3%A1tico-y-aprendizaje-profundo-862ca9790bb9

Cesar Juárez Megías, J. S. (2017). CLASIFICACIÓN DE IMÁGENES. 2-3.

Chas, A. (12 de Mayo de 2016). *Tipos de algoritmos de Inteligencia Artificial y Machine Learning*. Obtenido de AuraPortal BPM: https://www.auraportal.com/es/tipos-de-algoritmos-de-inteligencia-artificial-y-machine-learning/

Computational Learning Theory. (17 de 05 de 2019). *Computer Vsion*. Obtenido de DeepAI: https://deepai.org/machine-learning-glossary-and-terms/computer-vision

Corrigan, F. (30 de junio de 2019). *DroneZon*. Obtenido de DroneZon: https://www.dronezon.com/learn-about-drones-quadcopters/what-is-drone-technology-or-how-does-drone-technology-work/

CS231n . (16 de Octubre de 2018). *Convolutional Neural Networks for Visual Recognition*. Obtenido de https://cs231n.github.io/convolutional-networks/#fc

Das, S. (16 de Noviembre de 2017). *CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more*. Obtenido de Medium: https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Das, V. &. (1017). Redes Neuronales. 5-6.

DeepAI. (17 de 05 de 2019). *Machine Learning*. Obtenido de DeepAI: https://deepai.org/machine-learning-glossary-and-terms/machine-learning

Dertat, A. (8 de Agosto de 2017). *Aprendizaje Profundo Aplicado - Parte 1: Redes Neuronales Artificiales.* Obtenido de towardsdatascience:

https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6#04e7

Deshpande, A. (20 de Julio de 2016). *A Beginner's Guide To Understanding Convolutional Neural Networks*. Obtenido de https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

ElectronicWings. (8 de Enero de 2018). *ElectronicWings*. Obtenido de https://www.electronicwings.com/sensors-modules/xbee-module

Erdem, K. (6 de Febrero de 2019). *Understanding Region of Interest -(RoI Pooling).* Obtenido de Towards Data Science: https://towardsdatascience.com/understanding-region-of-interest-part-1-roi-pooling-e4f5dd65bb44

Fernando Sancho Caparrini. (20 de Abril de 2020). *Fernando Sancho Caparrini*. Obtenido de Aprendizaje Supervisado y No Supervisado: http://www.cs.us.es/~fsancho/?e=77

Franklin, D. (2016). *Computer vision is a field that includes methods for acquiring,prcessing, analyzing, and understanding images and, in general, high-dimensional data.* Obtenido de SlidePlayer: https://slideplayer.com/slide/10117716/

Frossard, D. (17 de Junio de 2016). Obtenido de VGG in TensorFlow: https://www.cs.toronto.edu/~frossard/post/vgg16/

Frossard, D. (17 de Junio de 2016). *VGG in TensorFlow*. Obtenido de https://www.cs.toronto.edu/~frossard/post/vgg16/

García, B. S. (2018). APLICACIÓN DE APRENDIZAJE PROFUNDO EN LA IDENTIFICACIÓN DE OBSTÁCULOS EN EL TRAYECTO DE VEHÍCULOS. 34-44.

García, L. (5 de Marzo de 2017). *Aprendizaje Profundo para Visión Artificial con MATLAB.* Obtenido de MathWorks: https://www.mathworks.com/videos/deep-learning-for-computer-vision-with-matlab-1540981496452.html?elqsid=1591909150641&potential_use=Student

Girshick, R. (25 de Septiembre de 2015). *Fast R-CNN.* Obtenido de Arxiv: https://arxiv.org/pdf/1504.08083.pdf

ImageNet. (2013). *ImageNet*. Obtenido de http://www.image-net.org/

Karen Simonyan, A. Z. (10 de Abril de 2015). *VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION.* Obtenido de https://arxiv.org/pdf/1409.1556.pdf

Ketkar, N. (2017). En N. Ketkar, *Deep Learning with Python* (pág. 226). A Hands-on Introduction: Apress.

Lucas García, M. (16 de Junio de 2016). *Aprendizaje Profundo para Visión Artificial con MATLAB*. Obtenido de MathWorks: https://www.mathworks.com/videos/deep-learning-for-computer-vision-with-matlab-1540981496452.html?elqsid=1591909150641&potential_use=Student

Malacara, D. (4 de 06 de 2019). *PROCESAMIENTO DE IMÁGENES*. Obtenido de ÓPTICA TRADICIONAL Y MODERNA: http://bibliotecadigital.ilce.edu.mx/sites/ciencia/volumen2/ciencia3/084/htm/optica.htm

MathWorks. (2017). *Deep learning approach to train new models faster by using pretrained models*. Obtenido de Transfer Learning: https://www.mathworks.com/discovery/transfer-learning.html

MathWorks. (2018). *CNN / Transfer Learning Example*. Obtenido de https://www.mathworks.com/matlabcentral/fileexchange/57280-cnn-transfer-learning-example

MathWorks. (7 de 2 de 2019). *Introducing Machine Learning.* Obtenido de MathWorks: https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/i/88174_92991v00_machine_learning_section1_ebook.pdf

MathWorks. (8 de Enero de 2019). *Machine Learning in MATLAB*. Obtenido de MathWorks: https://www.mathworks.com/help/stats/machine-learning-in-matlab.html

MathWorks. (2 de Mayo de 2019). *Machine Learning in MATLAB.* Obtenido de https://www.mathworks.com/help/stats/machine-learning-in-matlab.html

MathWorks. (23 de Mayo de 2019). *What Is a Convolutional Neural Network?* Obtenido de Deep Learning: https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

MathWorks. (17 de Marzo de 2019). *What Is Deep Learning?* Obtenido de Deep Learning: https://www.mathworks.com/discovery/deep-learning.html

MathWorks. (8 de Enero de 2019). *What Is Machine Learning?* Obtenido de https://www.mathworks.com/help/stats/machine-learning-in-matlab.html

MathWorks. (8 de Febrero de 2020). Obtenido de Object Detection Using Faster R-CNN Deep Learning: https://www.mathworks.com/help/vision/examples/object-detection-using-faster-r-cnn-deep-learning.html

MatWorks. (2016). *Train Object Detector Using R-CNN Deep Learning*. Obtenido de MatWorks: https://www.mathworks.com/help/vision/examples/object-detection-using-deep-learning.html

MyGica. (5 de Febrero de 2017). *MyGica*. Obtenido de https://www.mygica.com/product/capit/

Neurohive. (20 de Noviembre de 2018). *VGG16 – Convolutional Network for Classification and Detection*. Obtenido de https://neurohive.io/en/popular-networks/vgg16/

Nicholson, C. (20 de Febrero de 2018). *A Beginner's Guide to Neural Networks and Deep Learning*. Obtenido de pathmind: https://pathmind.com/wiki/neural-network#define

Olivera, O. G.-O. (16 de Septiembre de 2019). *Redes Neuronales artificiales: Qué son y cómo se entrenan.* Obtenido de xeridia: https://www.xeridia.com/blog/redes-neuronales-artificiales-que-son-y-como-se-entrenan-parte-i

PataganiaTec. (16 de Mayo de 2016). *Saber Sobre Arduino Pro Mini*. Obtenido de https://saber.patagoniatec.com/2014/07/arduino-pro-micro-5v-atmega32u4-micro-leonardo-pro-mini-arduino-argentina-ptec/

PNGIDS. (09 de Junio de 2017). *Ministerio del Ambiente*. Obtenido de Ministerio del Ambiente: http://www.ambiente.gob.ec/programa-pngids-ecuador/#

Rendon, J. J. (2018). Aprendizaje Profundo para la Identificación de Objetos en Robótica Móvil. 3-8.

Reyes, J. (2015). Construcción de una Interfaz Eléctrica de Salida para Sistema se Vsion Artificial. *ResearchGate*, 9.

Riofrío, J. L.-3. (06 de Marzo de 2017). *Instituto Nacional de Estadística y Censos*. Obtenido de Buenas cifras, mejores vidas: http://www.ecuadorencifras.gob.ec/estadisticas/

Rosembuj, T. (05 de 2018). *Inteligencia Artificial 1. Algoritmo*. Obtenido de elFisco: http://elfisco.com/articulos/1627

Santos, L. A. (23 de Marzo de 2018). *Rectified-Linear unit Layer*. Obtenido de artificial-inteligence: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/relu_layer.html

Shaoqing Ren, K. H. (6 de Julio de 2016). Faster R-CNN: Towards Real-Time Object. *Arxiv*, 5-9. Obtenido de arxiv: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

Sharma, P. (10 de 1 de 2019). *Construya su primer modelo de clasificación de imágenes*. Obtenido de AnalyticsVidhya: https://www.analyticsvidhya.com/blog/2019/01/build-image-classification-model-10-minutes/

T&D. (12 de Marzo de 2016). *T&Drones*. Obtenido de transmisor-video-fpv: https://www.tutorialdedrones.com/transmisor-video-fpv/

The data science and artificial intelligence. (17 de 05 de 2019). *Artificial Intelligence*. Obtenido de DeepAI: https://deepai.org/machine-learning-glossary-and-terms/artificial-intelligence

The MathWorks. (2019). *Artificial Intelligence (AI).* Obtenido de MATLAB for Artificial Intelligence: https://www.mathworks.com/discovery/artificial-intelligence.html

Tsang, S.-H. (24 de Agosto de 2018). *Review: GoogLeNet (Inception v1)— Winner of ILSVRC 2014 (Image Classification).* Obtenido de Medium: https://medium.com/coinmonks/paper-review-of-googlenet-inception-v1-winner-of-ilsvlc-2014-image-classification-c2b3565a64e7

Uijling J.R.R, V. d. (2013). Selective Search for Object Recognition. *fnwi*, 1-14.

Updated, L. (15 de Diciembre de 2019). *Residual Net.* Obtenido de Artificial-Inteligence: https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/deep_learning/residual_net

ANEXOS

ANEXO 1

Especificaciones Drone Mavic 2 Pro

CARACTERÍSTICAS	MAVIC 2 PRO	MAVIC 2 ZOOM
Dimensiones	91 x 214 x 84 mm	91 x 214 x 84 mm
Peso	907 gr	905 gr
Altura máxima vuelo	6.000 m sobre nivel del mar	6.000 m sobre nivel del mar
Batería	Intercambiable con autonomía máxima 31 minutos según condiciones de vuelo	
Velocidad de vuelo máxima	72 km/h en modo Sport	
Frecuencia de funcionamiento	2,4 y 5,5 Ghz (OcuSync 2.0)	
GNSS	GPS y Glonass	
Sistema de detección	Detección de obstáculos omnidireccional	
Cámara	Hasselblad Sensor 1" CMOS 20MP	Sensor 1/23" CMOS 12MP
Apertura	Variable F2.8/F11	F2.8 (28mm)/F3.8 (48mm)
Fotografía	Resolución máxima 5472 x 3648 (JPG y DNG)	Resolución máxima 4000 x 3000 (JPG y DNG)
Vídeo	4K a 24/25/30p, 2,7K a 24/25/30/48/50/60p y FHD a 24/25/30/48/50/60/120p	
Modos de color	DLog-M (10 bits), permite vídeo HDR (HLG 10 Bits)	D-Cinelike
Formato grabación	MP4/Mov (códec H264 y H265)	

ANEXO 2

Especificaciones FPV Camera 960H 2.5mm

Model	960H FPV Camera
Sensor	1/2.7' Mg SUPER HAD CCD
Horizotal Resoulution	D1 960H
Lens	2.8mm/2.5mm/2.1mm
Signal System	PAL/NTSC (OSD Internal adjustable)
S/N Ratio	>60Db(AGC OFF)
Electronic Shutter Speed	PAL:1/50-100.000,NTSC:1/60-100.000
Auto Gain Control	YES
Back Light Compensation	YES
Min Illumination	0.001Lux/1.2F
DNR	2DN
Power	DC 5V-23V
Weight	15g
Size	25mm*25mm
	OSD
exposure	Brightness/exposure mode/gain
The white balance	Automatic tracking white balance
Day and night mode	Internal automatic/color mode/black and white mode/digital wide dynamic
Video settings	Contrast/sharpness/saturation/digital noise/video format(N/P)
Language	Chinese/English/German/Italian/Russian
Restore the factory setting	OK
Save the exit	OK

ANEXO 3

Módulo de comunicación inalámbrica XBee S2C

Características

- Voltaje de operación 3.3v a 215mA
- Velocidad de transferencia 250kbps Max
- Potencia de salida 63mW(+17dBm)
- Alcance de 1500 metros aprox.
- Conector RPSMA para antena externa
- Encriptación 128-bit

Módulo XBee Series 2C

Fuente: (ElectronicWings, 2018)

ANEXO 4

Módulos de transmisión y recepción de video

Conjunto de transmisor y receptor de audio y video inalámbrico de 5.8 Ghz y 6000mW.

Características transmisor TS832

- Bandas de cubierta A, b, E, F, r 40 canales
- Formato de video NTSC/PAL
- Impedancia de salida 50 ohmios
- Distancia de transmisión 3000m aprox. (área abierta)
- Conector RP-SMA de antena

- Voltaje de funcionamiento 7-16v
- Corriente de suministro 220mA
- Impedancia de entrada de video 75 Ohm

Transmisor TS832 40Ch

Fuente: (T&D, 2016)

Características receptor RC832

- Impedancia de la antena 50 Ohm
- Impedancia de video 75 Ohm
- Frecuencia de trabajo 5.8 Ghz
- Canales disponibles 40 CH
- Consumo de corriente 200Ma
- Formato de video NTSC/PAL
- Voltaje de funcionamiento DC 12V
- Paso 85 g

Receptor RC832 40Ch

Fuente: (T&D, 2016)

ANEXO 5

Dispositivo de captura de video USB MyGica Capit para Windows

- Transferencia de datos USB 2.0 de alta velocidad para una mejor grabación de video
- Detección automática NTSC / PAL / SECAM
- Admite control de brillo, tono, contraste, saturación y nitidez
- Capture el video directamente en formato MPEG 1/2/4
- Fuente de Captura de vídeo desde VHS, V8 y Hi8, etc.
- Admite captura de imágenes fijas en formato JPEG o BMP

USB MyGica Capit

Fuente: (MyGica, 2017)

ANEXO 6

Código de Matlab para la detección y clasificación de objetos contaminantes en los *frames* de video.

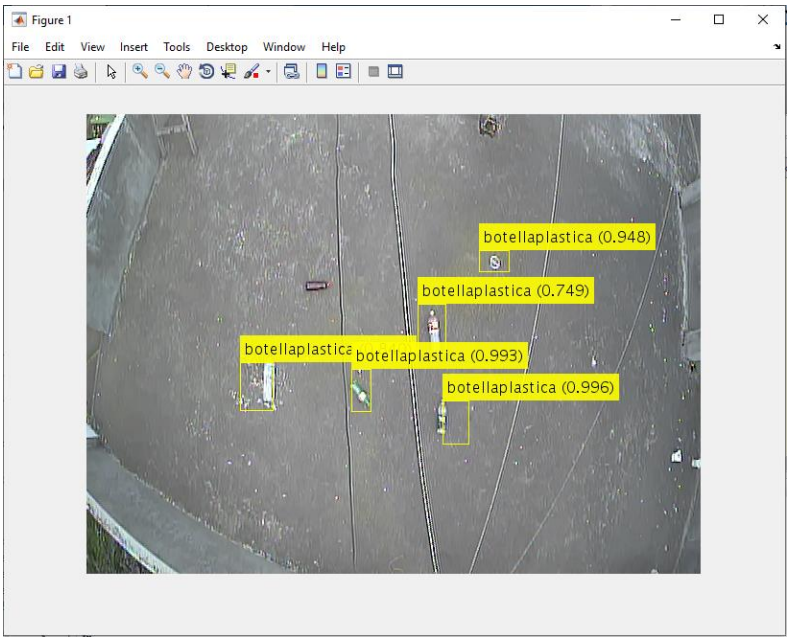
```
% Entrenamiento del detector de tipo FASTER R-CNN
%% Cargar los datos de entrenamiento
load('D:\Tesisimagenes\ObjetosContaminantes.mat');
ObjetosContaminatesDataset=objectDetectorTrainingData(gTruth,'SamplingFac
tor',1);

%% Separar los datos de entrenamiento y datos de prueba
idx = floor(0.7 * height(ObjetosContaminatesDataset));
trainingData = ObjetosContaminatesDataset (1:idx,:);
testData = ObjetosContaminatesDataset (idx:end,:);

%% Cargar la red pre-entrenada (AlexNet)
net = alexnet;
%Revisar la arquitectura de la red neuronal pre-entrenada
layers = net.Layers
% Modificar la red pre-entrenada
%filterSize = [9 9];
%numFilters = 96;
num_objects = size(trainingData,2)-1; %numero de categorias
%layers(1) = imageInputLayer([227 227 3]);
layers(23) = fullyConnectedLayer(num_objects,'Name','fc8');
layers(25) = classificationLayer('Name','myNewClassifier')

layers(end-2).WeightLearnRateFactor = 20;%10
layers(end-2).BiasLearnRateFactor = 20;%20

%%
%Options for step 1.
optionsStage1 = trainingOptions('sgdm',...
    'MaxEpochs',10,...
    'MiniBatchSize', 128,...
    'InitialLearnRate', 1e-3,...
    'CheckpointPath', tempdir);

%Options for step 2.
optionsStage2 = trainingOptions('sgdm',...
    'MaxEpochs',10,...
    'MiniBatchSize', 64,...
    'InitialLearnRate', 1e-3,...
    'CheckpointPath', tempdir);

%Options for step 3.
optionsStage3 = trainingOptions('sgdm',...
    'MaxEpochs',10,...
    'MiniBatchSize', 128,...
    'InitialLearnRate', 1e-3,...
    'CheckpointPath', tempdir);

%Options for step 4.
```

```
optionsStage4 = trainingOptions('sgdm',...
    'MaxEpochs',10,...
    'MiniBatchSize', 64,...
    'InitialLearnRate', 1e-3,...
    'CheckpointPath', tempdir);
options = [
    optionsStage1
    optionsStage2
    optionsStage3
    optionsStage4
    ];

% Inicialización
rng(0);
tic;
%Entrenar el detector
detectorfinal = trainFasterRCNNObjectDetector(trainingData, net, options,
...
    'NegativeOverlapRange', [0 0.3], ...
    'PositiveOverlapRange', [0.6 1], ...
    'SmallestImageDimension', [300], ...
    'BoxPyramidScale', 1.2);
toc;

%%--Prueba del detector en una imagen
% Read a test image
frame = imread('D:\tesisimagenes\Laguna50.jpg');
% Run the detector
[bbox, score, label] = detect(detectorfinal,I,'Threshold',0.85);
comp = isempty(label);
if comp == 1
    figure, imshow(I);
else
    if comp ==0
% Run nonmaximal suppression on the bounding boxes
[selectedBbox,selectedScore] = selectStrongestBbox(bbox,score,
'OverlapThreshold', 0.1); %'Union'
annotation=cell(size(selectedBbox,1),1);
for iii = 1:size(selectedBbox,1)
    annotation{iii} = sprintf('%s (%1.3f)', label(iii),
selectedScore(iii));
end
% Annotate detections in the image
I1 =
insertObjectAnnotation(frame,'rectangle',selectedBbox,cellstr(annotation)
,'Color','red','TextBoxOpacity',0.8,'FontSize',15);
figure, imshow(I1);
    end
end

%% Prueba del detector en los frames de video
% Obtener una imagen con la cámara fpv
vidObj = videoinput('winvideo', 1, 'UYVY_720x480');
src = getselectedsource(vidObj);
triggerconfig(vidObj ,'manual');
set(vidObj, 'TriggerRepeat',inf);
set(vidObj, 'FramesPerTrigger',1);
```

```
vidObj.FrameGrabInterval = 5;
vidObj.ReturnedColorspace = 'rgb';
vidObj.ROIPosition = [50 30 640 480];
start(vidObj);
pause(3); %pause(5)

load('FRCNNdetectorfinal.mat');
LagunaDetector = detectorfinal;

frameNumber = 0;
plastico = 0;
metal = 0;
totora = 0;
runLoop = true;
speed = 5;
while runLoop && frameNumber < 500

    pause(0.05);
    trigger(vid);
    for i = 1:speed
        frame = getsnapshot(vid);
        pause(0.3);
    end
    [bbox,score,label] = detect(LagunaDetector,frame,'Threshold',0.85);
    annotation=cell(size(bbox,1),1);
    for iii = 1:size(bbox,1)
        annotation{iii} = sprintf('%s (%1.3f)', label(iii), score(iii));

        if label(iii) == 'botellaplastica'
            plastico = plastico + 1;

        else
            if label(iii) == 'envasemetalico'
                metal = metal+1;

            else
                if label(iii) == 'totora'
                    totora = totora+1;

                end
            end
        end
    end
    if (~isempty(annotation))
    outputImage =
insertObjectAnnotation(frame,'rectangle',bbox,cellstr(annotation),'TextBo
xOpacity',0.9,'FontSize',20,'LineWidth', 3);
    else
        outputImage = frame;
    end
    imshow(outputImage);
end

stop(vid);
delete(vid)
```

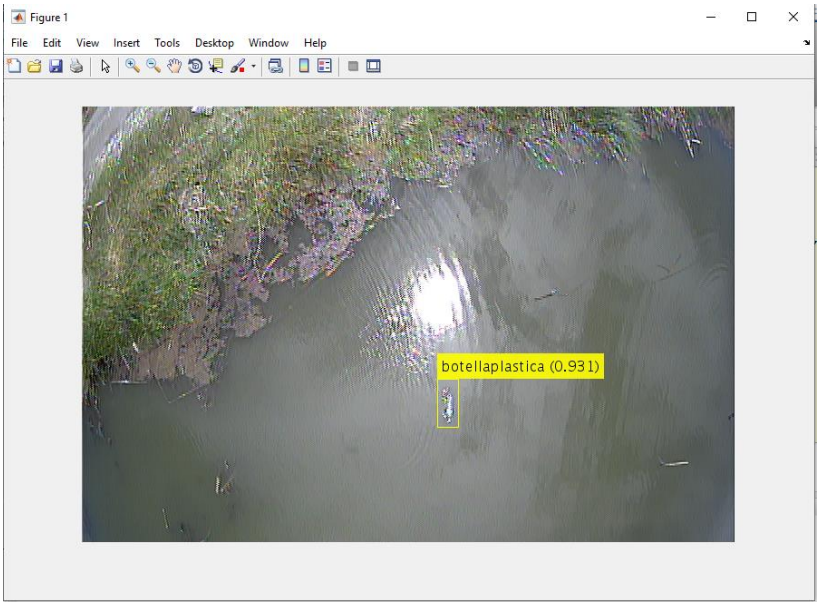
ANEXO 7

Entornos de prueba



Terraza de la Vivienda

Fuente: (Autor)



Laguna 1/2 km vía a Guano

Fuente: (Autor)