ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA TELECOMUNICACIONES Y REDES

"USO DE ALGORITMOS BASADOS EN MACHINE LEARNING PARA LA DETECCIÓN DE ENFERMEDADES EN PLANTAS FRUTALES DE PISO CLIMÁTICO TEMPLADO UTILIZANDO PROCESAMIENTO DE IMÁGENES CASO PRACTICO MANZANO."

Trabajo de titulación

Tipo: Propuesta Tecnológica

Presentado para optar al grado académico de:

INGENIERO EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

AUTOR: DIEGO SEBASTIÁN GORDÓN IZURIETA

DIRECTOR: ING. HUGO MORENO.

Riobamba - Ecuador

2021

Yo, **DIEGO SEBASTIÁN GORDÓN IZURIETA** declaro que el presente trabajo de titulación es de mi autoría y los resultados del mismo son auténticos. Los textos en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autor asumo la responsabilidad legal y académica de los contenidos de este trabajo de titulación; El patrimonio intelectual pertenece a la Escuela Superior Politécnica de Chimborazo.

Riobamba, 2021

Diego Sebastián Gordón Izurieta

1804420329

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES

El Tribunal del trabajo de titulación certifica que: El trabajo de titulación: Tipo: Propuesta Tecnológica; **"USO DE ALGORITMOS BASADOS EN MACHINE LEARNING PARA LA DETECCIÓN DE ENFERMEDADES EN PLANTAS FRUTALES DE PISO CLIMÁTICO TEMPLADO UTILIZANDO PROCESAMIENTO DE IMÁGENES CASO PRACTICO MANZANO."**, realizado por el señor **GORDÓN IZURIETA DIEGO SEBASTIÁN**, ha sido minuciosamente revisado por los Miembros del Tribunal del trabajo de titulación, el mismo que cumple con los requisitos científicos, técnicos, legales, en tal virtud el Tribunal autoriza su presentación.

	FIRMA	**FECHA**
ING. PABLO LOZADA **PRESIDENTE DEL TRIBUNAL**	Firmado digitalmente por PABLO EDUARDO LOZADA YANEZ	12/ marzo /2021
ING. HUGO MORENO **DIRECTOR DEL TRABAJO DE TITULACIÓN**		12/ marzo /2021
ING. VERONICA MORA **MIEMBRO DEL TRIBUNAL**	VERONICA ELIZABETH MORA CHUNLLO Digitally signed by VERONICA ELIZABETH MORA CHUNLLO Date: 2021.04.14 16:41:10 -05'00'	12/ marzo /2021

DEDICATORIA

El presente trabajo lo dedico a Dios y a mi padre que desde el cielo vela por mí. A mi madre Cecilia que me enseño que, con esfuerzo, sacrificio y mucho empeño se llegan a cumplir los sueños. A mi hermano Christian que siempre me apoyo en cada paso. Como olvidar a mis amigos Mauricio, Gabriel y Edison que con sus ocurrencias siempre me sacaron una sonrisa y me impulsaron a seguir adelante. A toda mi familia porque con sus oraciones, consejos y palabras de aliento me acompañaron en este arduo camino.

Diego

AGRADECIMIENTO

Estoy muy agradecido con la institución que me supo abrir sus puertas, la Escuela Superior Politécnica de Chimborazo y a cada uno de los maestros que aportaron en mi educación.

Mi sincero agradecimiento al doctor Hugo Moreno Avilés y a la ingeniera Verónica Mora, por su paciencia y su conocimiento para encaminar el presente proyecto.

Gracias a toda mi familia por ser siempre mi apoyo incondicional durante esta dura etapa. A mis amigos que fueron siempre un gran apoyo en cada etapa apoyándome incondicionalmente.

Diego

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ECUACIONES

ÍNDICE DE ANEXOS

ÍNDICE DE ABREVIATURAS

VANT	Un vehículo aéreo no tripulado
IoT	Internet of Things
CMOS	Semiconductor complementario de óxido metálico
RNA	Redes Neuronales Artificiales
AC	Antes de Cristo
IA	Inteligencia Artificial
tanh	Tangente
ART	Teoría de la resonancia adaptativa
C	Celsius
MATLAB	Laboratorio de matrices
LTE	Evolución a largo plazo
HOG	Histogram of Oriented Gradients
LBP	Local Binary Pattern
CPU	Central Processing Unit
RAM	Random Access Memory
SIG	Sistema De Información Geográfica
WiFi	Wireless Fidelity
CDMA	Code Division Multiple Access
GSM	Global System for Mobile
HSPA	High-Speed Packet Access
MMS	Multimedia Message Service
SMS	Short Message Service
UMTS	Universal Mobile Telecommunications System
Mp3	Moving picture experts group audio layer 3
Gif	Graphic Interchange Format
PNG	Portable Network Graphics
WAV	Waveform audio file format
MIDI	Musical Instrument Digital Interface

m	Metros
RDI	Regiones rectangulares de interés
PC	Computadora personal

RESUMEN

El presente trabajo de titulación tuvo como objetivo realizar la detección de enfermedades de la planta de manzano mediante la aplicación de Redes Neuronales Artificiales basadas en Machine Learning, seleccionando el entorno de procesamiento adecuado, para la extracción de las muestras en el terreno de estudio se utilizó un vehículo aéreo no tripulado con el cual se capturó varios videos de las plantas de manzano, posteriormente se realizó el procesamiento de los videos con la ayuda del software Matlab y se ejecutó la extracción de las muestras a partir de los videos, las cuales se almacenaron en una carpeta. Se inicio el proceso de entrenamiento de la red neuronal mediante el software Matlab, con la ayuda de la herramienta "*trainCascadeObjectDetector",* se incurrió en la selección del área de estudio con cada imagen, una vez hecho esto se realizó el entrenamiento con tres algoritmos de aprendizaje LBP, HOG, HAAR cada uno de ellos posee características únicas y con un tiempo determinado de procesamiento, ejecutado el entrenamiento de cada una de las enfermedades de manzano se realizó las pruebas respectivas para comprobar la validez de la red neuronal artificial, ya que se obtuvo los resultados se procedió a un análisis de toda la información recopilada mediante una plataforma de IoT en este caso la llamada "*ThingSpeak*" en la cual se marcó cada detección positiva, en la plataforma IoT se utilizó tres campos, cada uno con su enfermedad correspondiente adicional de una ventana con la ubicación en tiempo real del sitio de estudio y dos ventanas en las cuales se realiza una comparación entre cada una de las enfermedades de estudio, adicionalmente se creó un blog en el cual se agregó información sobre las afecciones del manzano, el link de este blog está vinculado con la plataforma IoT.

Palabras clave: < INTERNET DE LAS COSAS (IOT) >, < PLATAFORMA THINKSPEAK >, <REDES NEURONALES ARTIFICIALES >, <MACHINE LEARNING >, <ALGORITMOS DE APRENDIZAJE >, <PROCESAMIENTO DE IMÁGENES >, <MATLAB (SOFTWARE)>|.

SUMMARY

The current research work aimed to detect the apple plant diseases through the Artificial Neural Networks app based on Machine Learning with a suitable processing environment. An uncrewed aerial vehicle (UAV) was used to extract the samples in the field under study, and several videos of the apple plants were captured. Later, the videos were processed with Matlab software, and the samples extracted from the videos were stored in a folder. The neural network training process was carried out using Matlab software. With the train Cascade Object Detector tool, the study area was selected in each image. Once this was done, the training was supported with three learning algorithms LBP, HOG, HAAR; each one has unique characteristics, but each one needs a specific processing time. Once the training of each of the apple tree diseases had been carried out, the respective tests were accepted to check the artificial neural network's validity. After the results were found, all the information obtained was uploaded to an IoT ThingSpeak platform, where each positive detection was marked for a better analysis. Besides, three windows were added to the IoT platform, each with the corresponding disease. A window with the study site's real-time location and two windows allow a comparison between each of the diseases under study. Additionally, a blog was created in which information on apple tree diseases was uploaded. The link of this blog is linked to the IoT platform.

Keywords: <INTERNET OF THINGS (IOT)>, <THING SPEAK (IOT PLATFORM)>, <ARTIFICIAL NEURAL NETWORKS>, <MACHINE LEARNING>, <LEARNING ALGORITHMS>, <IMAGE PROCESSING>, <MATLAB (SOFTWARE)>.

INTRODUCCIÓN

ANTECEDENTES

Los sitios apropiados para la producción del manzano deben tener suelo con profundidad efectiva mayor a 80 cm, con buena estructura, adecuada actividad biológica, sin limitantes de drenaje, topografía que permita la utilización de maquinaria, disponibilidad de agua de riego sin contaminación de metales pesados ni coliformes fecales, clima que permita un correcto desarrollo del huerto de acuerdo a la variedad y porta-injerto elegido es decir suficiente frío en otoño invierno y acumulación térmica primavera-verano, lugares ubicados a pleno sol, bien ventilados y sin heladas tardías en primavera (Gárate, 2014 pág. 15).

Un aspecto muy importante a considerar, es el manejo de plagas, las plagas y enfermedades que afectan a los manzanos, cada una de ellas daña de manera específica al árbol causando pérdidas económicas hacia los productores (Belloch, 2017 pág. 34). Cada una de estas enfermedades tiene una distinción que se puede detectar, pero muchas veces debido al tamaño del árbol de manzano puede complicar la detección de las mismas para ellos utilizaremos una propuesta tecnológica (Belloch, 2017 pág. 35).

Un vehículo aéreo no tripulado (VANT), o comúnmente dron, es una aeronave que vuela sin tripulación, reutilizable, capaz de mantener de manera autónoma un nivel de vuelo controlado y sostenido, y propulsado por un motor de explosión, eléctrico o de reacción, un VANT tiene una amplia variedad de formas, tamaños, configuraciones y características, además un VANT se es usado como un instrumento que nos permite obtener datos e información que posteriormente requieren de un tratamiento y un uso práctico (MARTÍN, 2014 pág. 26).

El IoT o Internet of Things (Internet de las Cosas) está cambiando la manera en la que nos relacionamos con el mundo físico, este posee varias aplicaciones que pretende facilitarnos el uso de servicios en sectores como la hostelería, el comercio o la agricultura, esta última área de aplicación parecería errónea pero se puede usar para hacer seguimiento de magnitudes como la temperatura, humedad, luminosidad, presentar datos de las plantaciones y demás factores que pueden influir en la producción (Rose, y otros, 2015 pág. 4).

Machine Learning es una disciplina del campo de la Inteligencia Artificial que aprende del entorno en el que funciona y de los datos que maneja para adelantar la solución a un problema (Belloch, 2017 pág. 38). De ahí que se pueda emplear en cualquier empresa, conociendo bien la información que se maneja, se pueden simplificar o automatizar tareas, e incluso se puede llegar a predecir las situaciones y dificultades con las que se va a encontrar el negocio (Belloch, 2017 pág. 38).

La inteligencia artificial está transformando la sociedad a múltiples niveles, tocando de una forma u otra a prácticamente todos los niveles de población y sectores de actividad, aunque pudiera parecer que la inteligencia artificial, aún a través de sus múltiples ramas, es exclusiva de empresas con vínculos tecnológicos, se trata de una tecnología transversal, que se puede aplicar a prácticamente cualquier rama de trabajo. Incluso dentro del sector agrícola (Hochel, y otros, 2008 pág. 8).

La aplicación de la Inteligencia Artificial en estas áreas pone en evidencia la necesidad de que las empresas relacionadas con el sector agrícola no se queden atrás, la aplicación de la Inteligencia Artificial cambiará el entorno competitivo, la Inteligencia Artificial y las nuevas tecnologías impactaran en la mano de obra rediseñando el modo de trabajo y reemplazando trabajos con automatización, las empresas tendrán que contar con una estrategia dirigida a la fuerza laboral, integración de sistemas con nuevas tecnologías, desarrollo de soluciones cognitivas y, en definitiva, construcción de un nuevo ecosistema de producción (Hardy, 2001 pág. 9).

En Colima estado de México, basado en el principio de Machine Learning se desarrolló un sistema de información que procesa imágenes para la identificación de plagas en el cultivo de plátano como la Sigatoka Negra, Sigatoka Amarilla, el Moko del plátano y otras, que afectan su producción. Este trabajo se desarrolla en áreas de cultivo ubicadas en el municipio de Manzanillo, Colima (Román, y otros, 2018 pág. 65). En la implementación del sistema, se usaron drones aéreos equipados con cámaras que utilizan sensores CMOS que obtienen imágenes de alta resolución, estas imágenes fueron procesadas utilizando técnicas de procesamiento de imágenes (Román, y otros, 2018 pág. 65).

FORMULACIÓN DEL PROBLEMA

¿Es posible la identificación de plagas en los manzanos con algoritmos de Machine Learning?

SISTEMATIZACIÓN DEL PROBLEMA

Para dar solución al problema planteado, es necesario ir respondiendo durante el desarrollo del trabajo las siguientes interrogantes:

- ¿Cuáles son los conceptos fundamentales de Machine Learning para reconocimiento de imágenes?
- ¿Es posible la identificación de plagas o enfermedades con sistemas Maching Learing?
- ¿El rendimiento del sistema para identificar las plagas es factible?

JUSTIFICACIÓN TEÓRICA

La Inteligencia artificial es la creación de máquinas que puedan evitar comportamientos inteligentes en cambio el Machine Learning es una rama de la Inteligencia artificial que busca dotar a las máquinas de capacidades de aprendizaje (Calvo, y otros, 2018 pág. 32).

La técnica de Machine Learning qué se utiliza será redes neuronales ya que estás aprenden de forma jerárquica la información, la misma que se aprende por niveles donde las capas aprenden cosas muy concretas y entre más capas se añada nuestra red podrá llegar a diferenciar más cosas la gran ventaja de las redes neuronales es que no hay límite de capas (Matich, 2001 pág. 53).
Los vehículos aéreos no tripulados VANT hacen posible que profesionales de todo el mundo dispongan de la información que necesitan para identificar los problemas potenciales y mitigar las pérdidas en el sector agrícola, el uso de esta tecnología ofrece la posibilidad de mapear y examinar los cultivos y a su vez realizar un tratamiento a la información obtenida (MARTÍN, 2014 pág. 18).

El sistema desarrollado presenta un enfoque innovador porque incluye herramientas tecnológicas de vanguardia y las condiciones agronómicas particulares de las áreas de producción del estado de Colima, estos resultados obtenidos mejoran las investigaciones previas al considerar la detección de otros agentes patógenos utilizando técnicas de redes neuronales y LBP, estos resultados pueden aplicarse a otras áreas de cultivo con condiciones de producción similares. Detección de plagas, drones, procesamiento de imágenes, redes neuronales (Matich, 2001 pág. 55).

Por investigaciones similares con mismo principio se sabe que esta implementación puede ayudar a los extensos sembríos en la identificación de plagas y enfermedades, de igual manera ayudara brindar una respuesta rápida hacia una plaga. Esto dará como resultado que se logre salvaguardar el cultivo (Calvo, y otros, 2018 pág. 35).

Para evaluar si el sistema está obteniendo los resultados adecuados todos los datos obtenidos serán revisados y supervisados por un Ingeniero Agrónomo asegurando así que el proyecto funcione de la manera más óptima posible (Calvo, y otros, 2018 pág. 36).

JUSTIFICACIÓN APLICATIVA

El estudio y desarrollo de este proyecto tiene como interés facilitar la identificación de plagas y enfermedades en arboles de manzano mediante el procesamiento de imágenes aplicando Machine Learning, debido a que muchas veces el reconocimiento de estos padecimientos en los árboles de manzano no se aprecia desde el nivel del suelo para ello se plantea tomar las capturas de imágenes desde el aire con la ayuda de un VANT que será operado, en la Figura 1 se detalla el diagrama del proyecto.

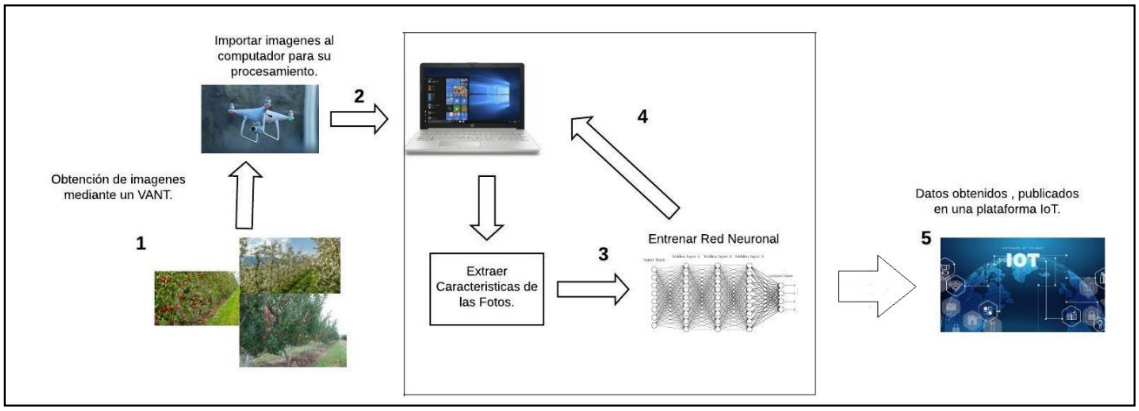


Gráfico 1-1: Diagrama de proyecto.

Fuente: (Diego Gordón, 2020).

En Figura 1 se detalla el diagrama del proyecto en el cual el paso 1 se captura imágenes de nuestra zona de estudio caso práctico manzanos con la ayuda de un VANT el cual será tele operado por toda la zona. Las mismas imágenes serán importadas en un pc la cual será encargada de realizar el procesamiento todo esto será el paso 2. El paso 3 y 4 será extraer las características de las fotos obtenidas por el VANT y entrenar a mi red neuronal todo este procesamiento lo realizará el pc luego de haber obtenido los datos necesarios estos serán publicados en una plataforma IoT.

OBJETIVOS

OBJETIVO GENERAL

- Usar algoritmos basados en machine learnig para la detección de enfermedades en plantas frutales de piso climático templado utilizando procesamiento de imágenes caso práctico manzano.

OBJETIVOS ESPECÍFICOS

- Verificar el funcionamiento de las redes neuronales artificiales, plataforma IoT y sus aplicaciones.
- Investigar las enfermedades más comunes que atacan a los árboles de manzano.
- Diseñar un sistema de identificación de plagas para árboles frutales de manzano basado en Machine Learning.

- Implementar en base a los algoritmos el rendimiento del sistema de identificación de plagas para árboles frutales de manzano.

➢ Realizar pruebas y obtener resultados de Machine Learning para la detección de enfermedades en plantas frutales de piso climático templado utilizando procesamiento de imágenes caso práctico manzano.

CAPÍTULO I

1. MARCO TEÓRICO

En el presente capítulo se analizan a las Redes Neuronales Artificiales (RNA), su arquitectura, su método de aprendizaje, así como también los beneficios que nos ofrece usar este tipo de redes para resolver problemas, luego se centra en su modo de aprendizaje y corrección de errores. Al final se detalla una serie de enfermedades de los manzanos.

1.1 Inteligencia Artificial

El hombre se ha aplicado a sí mismo el nombre científico de hombre sabio como una valoración de la trascendencia de sus habilidades mentales, tanto para la vida cotidiana como para el propio sentido de identidad, los esfuerzos de la inteligencia artificial, por su parte, se enfocan en lograr la comprensión de entidades inteligentes (Cruz, 2010 pág. 5). Por ello, una de las razones de su estudio y análisis es aprender acerca de los propios seres humanos, pero a diferencia de la filosofía y de la psicología, los esfuerzos de la inteligencia artificial están encaminados tanto a la construcción de entidades inteligentes, como a su comprensión, otra razón más por la que se estudia la inteligencia artificial es porque las entidades inteligentes así construidas son interesantes y útiles (Cruz, 2010 pág. 6).

Se podía considerar que uno de los primeros pasos hacia la Inteligencia Artificial fue dados hace mucho tiempo por Aristóteles (384-322 A.C), cuando se dispuso a explicar y codificar ciertos estilos de razonamiento deductivo que el llamo "Silogismos" (Cruz, 2010 pág. 6). Otro intento sería el de Ramón Llull, místico y poeta catalán, quien construyo un conjunto de ruedas llamado Ars Magna, el cual se suponía iba a ser una maquina capaz de responder todas las preguntas (Cruz, 2010 pág. 6).

En 1958, Jhon McCarthy, responsable de introducir el término "Inteligencia Artificial", propuso utilizar el cálculo proposicional como un idioma para representar y utilizar el conocimiento en un sistema que denominó la "Advice Taker" (Cruz, 2010 pág. 7). A este sistema se le tenía que decir que hacer en vez de ser programado (Cruz, 2010 pág. 7).

En los años setenta se inicia el desarrollo de aplicaciones de Inteligencia Artificial en términos de sistemas basados en reglas a los que se les llama primero "Sistemas Expertos" y después adoptan el nombre de "Sistemas Basados en Conocimiento". El reconocimiento de la carencia de la lógica

como herramienta única de representación da lugar al desarrollo de otras formas de representación e inferencia mediante redes causales y asociativas (semánticas, neuronales y bayesianas) y marcos, objetos y agentes (Méndez, y otros, 2008 pág. 209).

En la etapa dominada por el paradigma simbólico (1956-1986) hay un número claro de trabajos basados en los principios de la cibernética y el conexionismo, tales como la teoría modular de autómatas probabilísticos, las memorias asociativas, el reconocimiento de caracteres, los sistemas auto organizativos y la traducción automática, es en el año de 1986 cuando se retoma con fuerza la aproximación a redes neuronales al problema de la inteligencia artificial. En la década de los 90 aparece también, en el contexto de la robótica, el interés por la aproximación situada, cerrando así el lazo histórico que comenzó su desarrollo en el año de 1943, con el reconocimiento de la necesidad de usar los tres paradigmas tanto en la perspectiva teórica como en la aplicada (Méndez, y otros, 2008 pág. 209).

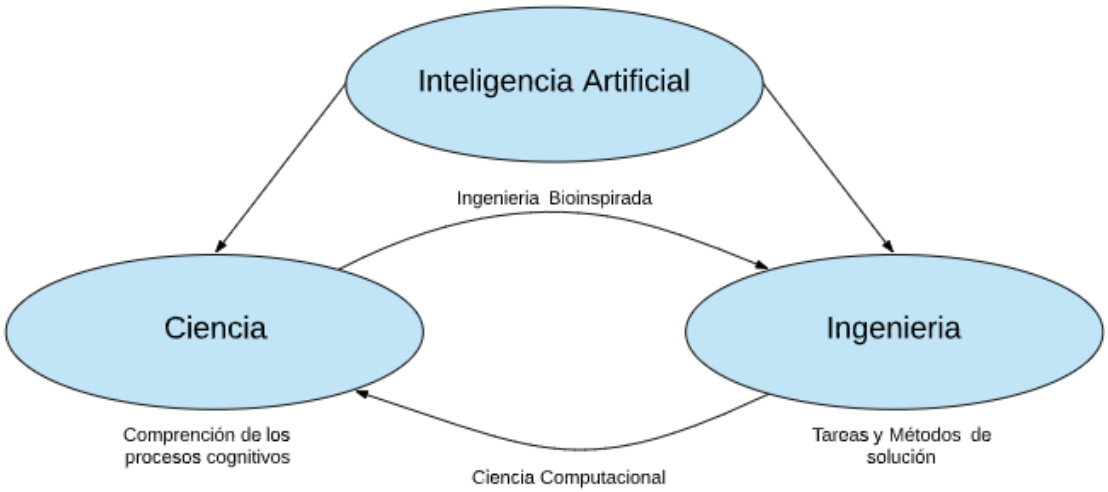


Gráfico 2-1: Relación de la inteligencia artificial con otras áreas.

Fuente: (Diego Gordón, 2020).

Si queremos que el conocimiento sea reutilizable, debe de ser impersonal en todo sentido, transferible, verificable experimentalmente y con la misma capacidad de predicción de una ley física actual. Desafortunadamente, no estamos seguros de disponer de las matemáticas necesarias para formalizar el conocimiento de los procesos cognitivos de la misma forma que la física dispone del cálculo diferencial e integral (Méndez, y otros, 2008 pág. 210).

Un aspecto importante para entender la parte computable de la inteligencia humana es detallar el conocimiento que se queda fuera cuando vamos atravesando fronteras de dominios y niveles en el proceso de reducción de un modelo conceptual, primero a un modelo formal y después a un programa (Méndez, y otros, 2008 pág. 210).

Figura 1-1: Conferencia sobre Inteligencia Artificial en Dartmouth College". Los próximos cincuenta años".

Fuente: (GUEVARA, y otros, 2008 pág. 58)

1.2 Redes Neuronales

W.S. McCulloch y W. Pitts introducen el concepto de neurona formal que ha dado origen al paradigma conexionista. Posteriormente K. Craik introduce los fundamentos del paradigma simbólico al interpretar el conocimiento humano en términos de descripciones declarativas y modulares de entidades simbólicas de alto nivel y de un conjunto de reglas inferenciales usadas para manipular esas descripciones simbólicas. Finalmente, en el tercer trabajo, A, Rosenblueth, N. Wiener y J. Bigelow introducen las bases del paradigma situado al interpretar el comportamiento inteligente en términos de un conjunto de mecanismos de realimentación. Posteriormente, el trabajo de la escuela de W.S. McCulloch, incluyendo la relevante aportación de J. von Neumann, crea las bases de la visión cibernética de la IA, que busca los fundamentos de la inteligencia en la enorme red de mecanismos genéticos, moleculares, neuronales, metabólicos y sociales de los que emerge. El trabajo de A. Turing en 1950 complementa los fundamentos del paradigma simbólico al proponer un procedimiento experimental de medir la inteligencia contenida en un "programa de IA" (lo que hoy conocemos como "test de Turing") (Méndez, y otros, 2008 pág. 2010).

No es absolutamente necesario creer que los modelos de redes neuronales tienen algo que ver con el sistema nervioso, pero ayuda, a su mejor entendimiento ya que seremos capaces de utilizar un gran cumulo de ideas, experimentos y hechos de las ciencias cognitivas y las neurociencias para diseñar, construir y probar las redes (Anderson, 2007 pág. 216).

Una red neuronal está constituida principalmente por nodos, o unidades, que están conectadas entre sí, a cada conexión se le asigna un peso numérico, estos pesos constituyen el principal recurso de memoria de largo plazo en las redes neuronales, y el aprendizaje se da con la actualización de los pesos (Russell, y otros, 1996 pág. 599). Algunas de las unidades están conectadas a un ambiente externo, estas pueden ser designadas como unidades de entrada o unidades de salida, los pesos son modificables de tal manera que la conducta de la entrada o salida de la red este más acorde a la realidad del proceso (Russell, y otros, 1996 pág. 599).

El elemento básico de una red neuronal es la neurona, la neurona biológica tiene 3 partes principales, las dendritas, el cuerpo de la neurona o también llamado Soma y el axón (Martinsanz, y otros, 2011 pág. 100). Las dendritas son fibras que transmiten las señales eléctricas al cuerpo de la neurona, el cuerpo de la neurona provocara un estímulo que transitara el estado de la neurona a las otras neuronas con las que está conectada, todas las conexiones son denominadas axones (Martinsanz, y otros, 2011 pág. 100).

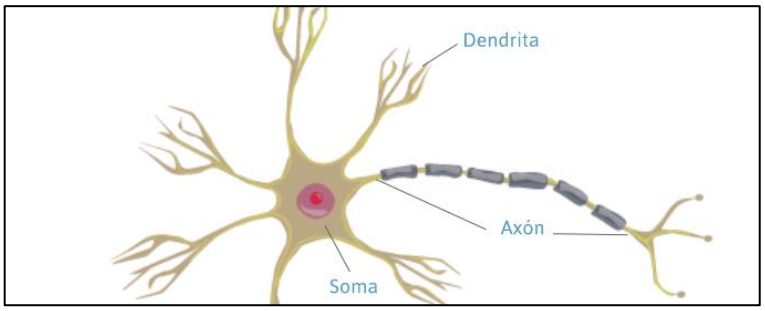


Figura 2-1: Composición de una red Biológica
Fuente: (Martinsanz, y otros, 2011 pág. 100).

Uno de los problemas más antiguos que la ciencia experimental a tratado de resolver es encontrar funciones que permitan el ajuste, o expliquen, datos que se observan de fenómenos naturales. La principal ventaja de la existencia de dichas funciones es la posibilidad de predecir el comportamiento del sistema natural en el futuro y controlar sus salidas mediante la aplicación de las entradas adecuadas a dicho sistema (Méndez, y otros, 2008 pág. 212).

1.2.1 Ventajas de las redes neuronales

Las ventajas de las redes neuronales artificiales, como sistemas de aprendizaje, son diversas, las más destacables son las que se relacionan seguidamente:

- Aprendizaje Adaptativo. - Posee la capacidad de aprender a realizar varias tareas basadas en la experiencia a partir de un entrenamiento con datos parciales (Martinsanz, y otros, 2011 pág. 104).

- Autoorganización. - La red puede crear su propia organización o su propia representación de la información después de una etapa de aprendizaje. Esta representación se basa en los datos utilizados para el aprendizaje (Martinsanz, y otros, 2011 pág. 105).

- Tolerancia a fallos. - Una destrucción parcial de la red conduce a una degradación de los resultados, pero en muchos casos puede seguir funcionando, pero se debe tener en cuenta que la salida puede llevar resultados erróneos de algún modo (Martinsanz, y otros, 2011 pág. 105).

- Operación en tiempo real. - Las redes pueden ser realizadas para trabajar en paralelo con otras áreas no solo de ingeniería (Martinsanz, y otros, 2011 pág. 105).

- Fácil inserción dentro de la tecnología existente. - Una red puede ser rápidamente entrenada, comprobada verificada y trasladada a Hardware con un reducido costo lo que permite su inserción para tareas específicas en diferentes áreas (Martinsanz, y otros, 2011 pág. 105).

- No linealidad. - La neurona realiza en general una transformación no lineal en sus entradas la combinación de las distintas neuronas podrá representar funciones no lineales tanto suaves como abruptas según se requiera (Martinsanz, y otros, 2011 pág. 105).

1.2.2 Desventajas de las redes neuronales

- Las redes neuronales se deben entrenar para cada problema. Además, es necesario realizar múltiples pruebas para determinar la arquitectura adecuada. El entrenamiento es largo y puede consumir varias horas de computo (Cruz, 2010 pág. 7).

- Debido a que las redes se entrenan en lugar de programarlas, estas necesitan muchos datos (Cruz, 2010 pág. 7).

- Las redes neuronales representan un aspecto complejo para un observador externo que desee realizar cambios. Para añadir nuevo conocimiento es necesario cambiar las iteraciones entre muchas unidades para que su efecto unificado sintetice este conocimiento. Para un problema de tamaño considerable es imposible hacer esto manualmente, por lo tanto, una red con representación distribuida debe emplear al esquema de aprendizaje (Cruz, 2010 pág. 7).

En ocasiones será posible desarrollar un modelo matemático exacto que explique el proceso del que provienen los datos que estamos obteniendo, sin embargo, no podremos dar detalles de ese

proceso, para lo cual el objetivo, en este caso, será estimar el modelo subyacente que genera los datos que están siendo observados. Las técnicas de aprendizaje automático nos permiten establecer modelos utilizando datos de ejemplo o experiencias pasadas. De este modo se podrán identificar patrones o ciertas regularidades en los datos y así podremos construir buenas aproximaciones al problema. Dentro de estas técnicas, que podríamos llamar de estimación semiparamétrica, se encuentran las Redes de Neuronas Artificiales (Méndez, y otros, 2008 pág. 212).

Una Red de Neuronas Artificiales es un modelo de procesamiento de información inspirado en el modo en el que lo hace el cerebro humano. El elemento clave de este modelo es su estructura. Las redes neuronales artificiales están compuestas por un cierto número de elementos de procesamiento o denominadas neuronas que trabajan al unísono con el fin de resolver un problema específico (Méndez, y otros, 2008 pág. 215).

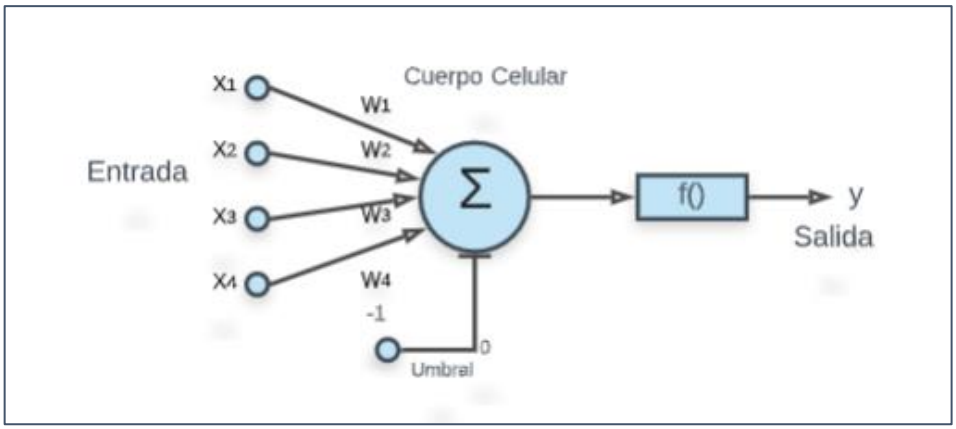


Figura 3-1: Modelo neuronal de McCulloch-Pitts.
Fuente: (Diego Gordón, 2020).

1.2.3 Elementos de una red neuronal

Existen varios elementos y parámetros que caracterizan una red neuronal como son el número de neuronas por cada capa, el grado y tipo de conectividad entre neuronas, el algoritmo de aprendizaje de la red, la función de activación, la función de salida el umbral y varios elementos más (Martinsanz, y otros, 2011 pág. 100).

1.2.3.1 Conexiones entre neuronas.

Las conexiones entre neuronas son el equivalente a los axones que transmiten la señal desde el cuerpo de una neurona hacia otras neuronas biológicas. El peso de una conexión entre dos neuronas vendrá dado por el peso que se le asigne a dicha conexión, los enlaces entre neuronas pueden ser excitadores o inhibidores respectivamente (Martinsanz, y otros, 2011 pág. 101).

1.2.3.2 Potencial de Entrada.

Una neurona se activa cuando el potencial sináptico alcanza un determinado umbral. Ese potencial es la suma de los potenciales que le llegan por las distintas conexiones de la red neuronal (Martinsanz, y otros, 2011 pág. 101).

1.2.3.3 Estado de activación.

El estado en el que se encuentra toda neurona (excitación o reposo), en las neuronas artificiales no se considera estado anterior por lo que el estado de activación dependerá exclusivamente de la entrada actual este valor puede ser discreto o continuo (Martinsanz, y otros, 2011 pág. 101).

1.2.3.4 Valores Discretos.

Estos valores pueden ser binarios, por lo general (-1,1) o (0,1) o incluso un conjunto de valores con un número pequeño de elementos, el valor 1 suele representa un estado de activo de la neurona y el 0 un estado pasivo de la neurona (Martinsanz, y otros, 2011 pág. 101).

1.2.3.5 Valores Continuos.

Estos valores continuos, pueden ser limitados o ilimitados (puede tomar cualquier valor real), un buen ejemplo de valores continuos limitados es cuando el estado de activación puede tomar un valor de (0,1) o incluso (-1,1) generalmente siendo una función sigmoidea o similar (Martinsanz, y otros, 2011 pág. 101).

1.2.3.6 Funciones de Activación.

El primer paso para poder obtener la salida y de la neurona es calcular la suma ponderada a de las entradas, llamada activación de la neurona esto de logra de la siguiente manera:

$$a = \sum_{i=1}^{D} w_i x_i + w_0$$

Ecuación 1-1: Función de Activación

Donde w_0 es un umbral o sesgo que se utiliza para compensar la diferencia entre el valor medio de las entradas, sobre todo el conjunto de entrenamiento, y el correspondiente valor medio de las salidas deseadas (Martinsanz, y otros, 2011 pág. 102). Posteriormente, a partir de este valor a se obtiene la salida y de la neurona mediante la aplicación de una función, llamada función de activación o de transferencia g(a), es decir:

$$y = g(a) = g\left(\sum_{i=1}^{D} w_i x_i + w_0\right) = g\left(\sum_{i=0}^{D} w_i x_i\right)$$

Ecuación 1-2: Función de Transferencia.

Donde, como se puede observar, es posible tratar el umbral w_0 como un peso más si se supone una entrada añadida x_0 con un valor fijo de 1. Finalmente, también es posible reescribir toda la ecuación en notación vectorial como g(a) = g (w^T x), si tomamos **w** como el vector de pesos y **x** como el vector de entradas a la red (Martinsanz, y otros, 2011 pág. 103).

La función de transferencia empleada en el modelo básico de McCulloch-Pitts es la función escalón definida por la siguiente ecuación:

$$g(a) = \begin{cases} 0 & cuando \quad a < 0 \\ 1 & cuando \quad a > 0 \end{cases}$$

Ecuación 1-3: Función de Transferencia de McCulloch-Pitts.

En los modelos actuales se escogen otro tipo de funciones, normalmente monótonas y derivables. Como la función lineal.

$$g(a) = a$$

Ecuación 1-4: Función lineal.

La sigmoidea

$$g(a) = \frac{1}{1 + e^{-a}}$$

Ecuación 1-5: Función sigmoidea.

la tangente hiperbólica (tanh)

$$g(a) = tanh = \frac{e^{a} - e^{-a}}{e^{a} + e^{-a}}$$

Ecuación 1-6: Tangente hiperbólica.

Y la gaussiana

$$g(a) = exp\left(-\frac{(a-u)^2}{2\sigma^2}\right)$$

Ecuación 1-7: Función Gaussiana.

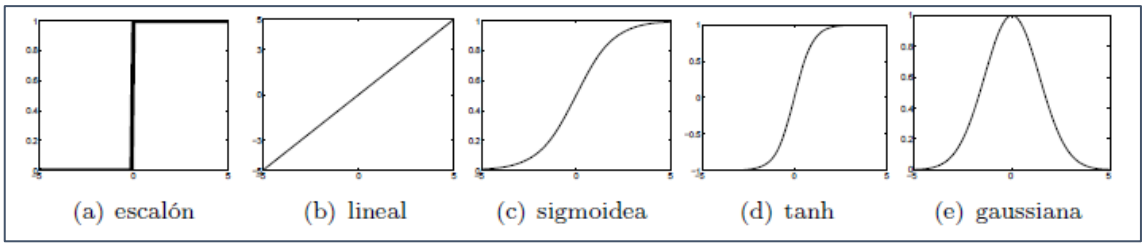


Figura 4-1 Principales funciones de transferencia utilizadas en redes neuronales.

Fuente: (Méndez, y otros, 2008 pág. 13)**.**

1.2.3.7 Niveles de neuronas y conectividad.

Las neuronas en una red neuronal se organizan por niveles o capas, por ello existen redes monocapa y redes multicapa (Martinsanz, y otros, 2011). Las neuronas se pueden conectar entre neuronas de una capa y la siguiente de modo de modo que no existan lazos entre ellas, un buen ejemplo de este tipo de redes son las del tipo hacia adelante (feedforward), también se permite que existan conexiones entre neuronas de la misma capa, conexiones hacia capas anteriores o incluso conexiones a sí misma, dependiendo del tipo de red (Martinsanz, y otros, 2011 pág. 102).

Conexiones solo hacia delante o también llamadas feedforward, no existen conexiones hacia atrás de una neurona, ni laterales (Martinsanz, et al., 2011). El Perceptrón, Adaline o el Perceptrón multicapa es uno de los ejemplos más claros de este tipo de conexión (Martinsanz, et al., 2011 p. 102).

Conexiones hacia atrás (feedback), suelen aparecer por lo general en redes bicapa y son útiles para realizar una asociación de la información de entrada con la información de salida, el mejor ejemplo de este tipo de redes son la ART o el Neocognitron (Martinsanz, et al., 2011 p. 102).

1.2.3.8 Tipo de asociación de entrada/ salida.

Según la relación que se cree entre los datos de entrada y su salida, las redes se pueden clasificar en heteroasociativas o autoasociativas (Martinsanz, et al., 2011 p. 102).

1.2.4 Tipos de redes

1.2.4.1 Redes heteroasociativas.

La red aprende parejas de datos [(A_1, B_1); (A_1, B_1); (A_1, B_1); ...] de forma que si se presenta A_1 la red responderá B_1 (Martinsanz, et al., 2011 p. 105). Este tipo de redes necesita al menos dos capas una para retener información de entrada y otra para mantener la salida (Martinsanz, et al., 2011 p. 106).

1.2.4.2 Redes autoasociativas.

Son aquellas en las que la red aprende ciertas informaciones A_1, A_2, A_3, A_4…, A_n de forma que cuando se presenta una información de entrada, se realiza una correlación respondiendo con uno

de los datos almacenados, el más que se le parezca (Martinsanz, et al., 2011). Su principal misión es reconstruir la información que se le incompleta o distorsionada (Martinsanz, et al., 2011 p. 104).

1.3 Tipos básicos de problemas de las redes neuronales

Existen dos tipos básicos de problemas que una RNA puede tratar de resolver: problemas de clasificación y problemas de regresión, en el primer tipo de problemas, el objetivo consiste en crear un procedimiento mediante el cual un nuevo caso representado por unos atributos observados o características que constituyen los datos de entrada a la RNA, se asigne a una de entre un conjunto de clases predefinidas, a la construcción de estos clasificadores también se le conoce como reconocimiento de patrones (Méndez, y otros, 2008 pág. 215). Un ejemplo de problemas de este tipo sería el reconocimiento de rostros o clasificación de imágenes, en este caso, la entrada a la RNA es una imagen de una persona natural, las clases son las personas a reconocer, y el programa de aprendizaje deberá aprender a asociar imágenes de caras a las identidades o clasificar las imágenes según patrones correspondientes a cada caso (Méndez, y otros, 2008 pág. 215).

Por otro lado, los problemas de regresión son problemas de ajuste de funciones, es decir, se trata de obtener un número en función de los atributos de entrada a la red o, lo que es lo mismo, se trata de encontrar una función continua de ciertas variables (Méndez, y otros, 2008 pág. 216). Un ejemplo de este tipo de problemas sería, por ejemplo, poder predecir la evolución de los consumos energéticos de un país a lo largo del año en función de ciertos datos históricos, época del año, etcétera (Méndez, y otros, 2008 pág. 216).

1.4 Proceso de aprendizaje de las redes neuronales

Una de las características más sobresalientes de las redes neuronales es su capacidad de aprendizaje, es decir, su disposición a modificar sus parámetros de forma relativamente sencilla a la interacción con el medio que las rodea, sea cual sea la forma de adaptación nos referiremos a un determinado tipo d aprendizaje u otro (Martinsanz, y otros, 2011 pág. 105).

El proceso de aprendizaje implica una serie de estímulos o interacciones de la red con el medio y un ajuste de sus parámetros mediante un algoritmo concreto, la interacción de la red con el medio se la conoce como paradigma o tipo de aprendizaje, entre los más destacados tenemos: aprendizaje supervisado, aprendizaje no supervisado, aprendizaje por refuerzo (Martinsanz, y otros, 2011 pág. 106).

Los algoritmos de aprendizaje se usan para ajustar los parámetros de la red podemos destacar los siguientes: corrección de error, Boltman, la regla de Hebb o el algoritmo competitivo (Martinsanz,

y otros, 2011 pág. 106). Para determinar que una red a aprendido los parámetros que se están ajustando deben permanecer estables durante la fase de entrenamiento a partir de un cierto momento, si los parámetros variables de la red son los pesos W_{ij} que conectan dos neuronas, se considera que a red a aprendido cuando:

$$\frac{dw_{ij}}{dt} = 0$$

Ecuación 1-8: Regla de Hebb.

1.4.1 Aprendizaje supervisado y no supervisado

Los sistemas más comunes de redes neuronales tienen procesos de aprendizaje similares, los cuales se basan en usar algoritmos de aprendizaje supervisado (Anderson, 2007 pág. 218). Para usar este algoritmo de aprendizaje tenemos un conjunto de datos y un conjunto de entrenamiento, para probar que estos sistemas estén funcionando adecuadamente debemos conocer la salida apropiada del sistema, esta puede ser correcta o equivocada. Dado un miembro del conjunto de entrenamiento como entrada de la red, podríamos saber cuál debería ser la salida y como difiere de lo que en realidad obtenemos, entre mayor información tengamos sobre el conjunto de entrenamiento y la red, mejor podremos aprender (Anderson, 2007 pág. 218).

Las aplicaciones donde este tipo de aprendizaje es más utilizado son en la clasificación de patrones, la aproximación de funciones y la identificación o construcción de modelos (Martinsanz, y otros, 2011 pág. 109).

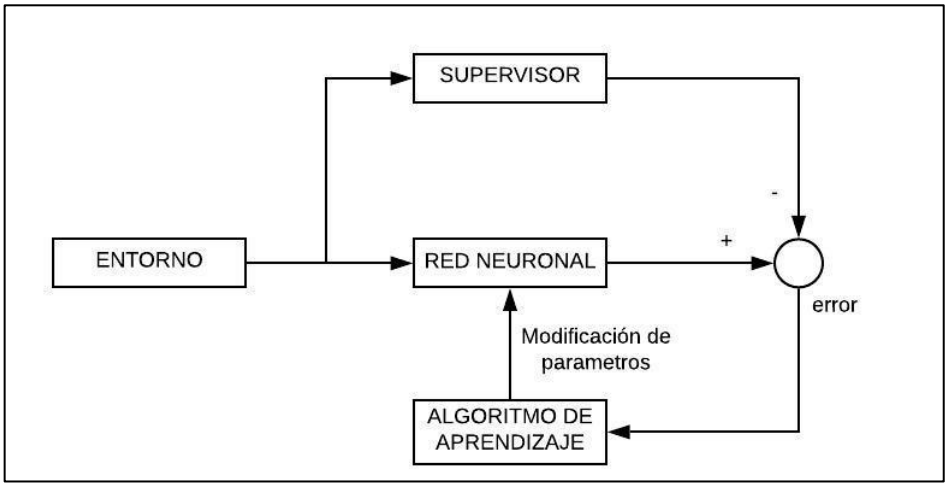


Gráfico 3-1: Esquema del tipo de aprendizaje supervisado

Fuente: (Diego Gordón, 2020).

También hay una clase de algoritmos en los cuales no existe un supervisor omnisciente, a este tipo se los llama algoritmos no supervisados son de gran interés, porqué examinan los datos de entrada y de alguna manera, organizan la información de una manera apropiada (Martinsanz, y otros,

2011 pág. 111). Esto casi parece mágico, pero en realidad en este tipo de algoritmos no existe nada de magia, pero como tienen poca información inicial para trabajar, los algoritmos no supervisados son difíciles de construir y de utilizar, generalmente hacen supuestos sobre lo que están tratando de hacer (Martinsanz, y otros, 2011 pág. 111).

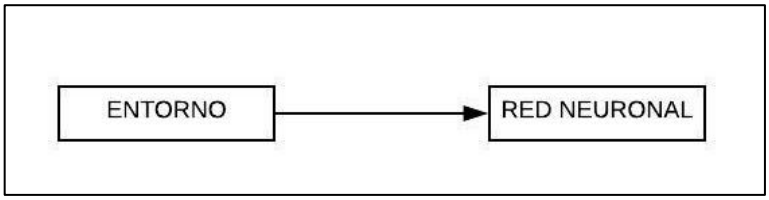


Gráfico 4-1: Esquema del tipo de aprendizaje no supervisado.
Fuente: (Diego Gordón, 2020).

1.4.2 Corrección de errores

Se conoce como error a la diferencia que existe entre la salida que proporciona la red y la salida que se espera que proporcione la misma red. Modificar los parámetros de la red para que disminuya este error es la forma más evidente de intentar que la red aprenda (Martinsanz, y otros, 2011 pág. 106).

Si consideramos a $y_n(k)$ la salida de una neurona n en el instante k y $d_n(k)$ la salida deseada, el error en el instante k para esa neurona se puede definir como:

$$e_n(k) = d_n(k) - y_n(k)$$

Ecuación 1-9: Corrección de errores.

El objetivo principal de la corrección será minimizar el error en todas las neuronas de salida. Para lograr ello se puede definir una función de coste en función del error de la red. Una vez hecha la función de coste, el problema del aprendizaje se reduce a uno de optimización (Martinsanz, y otros, 2011 pág. 107).

1.4.3 La regla de HEBB

La primera regla para este tipo de sistemas de aprendizaje fue propuesta por D. Hebb en 1949 [Hebb, 1949], y se basa en un hecho biológico constatado: cuando dos neuronas se activan simultáneamente su conexión se refuerza (Méndez, y otros, 2008 pág. 218). Esto es, cuando una o varias neuronas tiene un nivel positivo de actividad y se sienten estimuladas por otra neurona, tiende a producirse un refuerzo en la conexión que las enlaza. Por el contrario, si una neurona tiende a inhibir a otra, la fuerza de conexión entre ambas también tiende a disminuir (Méndez, y otros, 2008 pág. 218).

Todo esto se expresa matemáticamente de la siguiente manera:

$$W_i(\tau + 1) = W_i(\tau) + nx_i y$$

Ecuación 1-10: La regla de HEBB.

Donde, W_i en un determinado instante del entrenamiento $\tau + 1$, es igual al peso en el instante anterior más un incremento o decremento (Méndez, y otros, 2008 pág. 218). La magnitud de dicho cambio viene determinada tanto por la entrada x_i y la salida y de la neurona como por η conocida como ratio o tasa de aprendizaje, esto se puede observar que esta regla de aprendizaje es no supervisada, ya que se emplea la salida de la neurona, y, pero no la salida deseada, t (Méndez, y otros, 2008 pág. 218).

1.5 Tipos de redes neuronales artificiales

1.5.1 El Perceptrón

Las redes de prealimentación con niveles fueron estudiadas por primera vez a finales de la década de los 50 bajo el nombre de perceptrónes. Se sometieron a estudio redes de todos los tamaños y topologías, el único elemento de aprendizaje efectivo en esa época fueron las redes de un solo nivel, ya que en ellas fue concentrada la mayor parte de la atención (Russell, y otros, 1996 pág. 605).

El Perceptrón es un modelo simple de neurona que nos va a permitir presentar los conceptos básicos, al igual que una neurona real, a nuestro Perceptrón llegaran señales de entrada y saldrá una salida (SERRANO, y otros, 2012 pág. 45). Las entradas (x) del Perceptrón podrán tomar valores entre 0 y 1, además a cada una de las entradas se le asignara un valor el cual se denomina peso (w), la salida (z) es una función que dependerá del valor de las entradas, de sus pesos, y del umbral de disparo de la neurona (θ) y su valor está entre 0 y 1 (SERRANO, y otros, 2012 pág. 45).

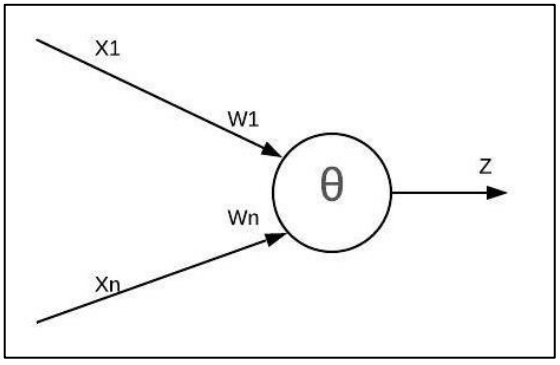


Figura 5-1 Esquema básico de un Perceptrón.

Fuente: (Diego Gordón, 2020).

La arquitectura definida por Rosenblatt definió para el Perceptrón consistía en una primera capa de j neuronas con funciones φj que se encargaban de transformar los datos de entrada (Méndez, y otros, 2008 pág. 218). Estas funciones reciben un subconjunto aleatorio de entradas a través de unos

pesos fijos y les aplican una función de activación de tipo escalón, existe un peso especial o sesgo w0 y, de igual modo que en la siguiente ecuación que se presenta, una entrada artificial x0 con valor 1 asociado a estepeso, en el caso del perceptrón se considera una función de activación extra φ0=1. La salida de esta primera capa de funciones φj se conecta a través de unos pesos a una última neurona para, finalmente, calcular la salida del perceptrón (Méndez, y otros, 2008 pág. 218).

$$y = g(a) = g\left(\sum_{j=0}^{M} W_j \emptyset_j(x)\right) = g(w^T \emptyset)$$

Ecuación 1-11: El Perceptrón.

Donde $\emptyset$ es el vector formado por las funciones de activación $\emptyset_0, \dots\dots, \emptyset_M$ y g es la función escalón definida para los valores $\{-1, 1\}$.

Una de las principales aportaciones del perceptrón es que la adaptación de sus pesos se realiza teniendo en cuenta el error entre la salida que obtiene la red y la salida que se desearía obtener es por ello que es considerado el primer método de aprendizaje supervisado (Méndez, y otros, 2008 pág. 219). Los perceptrones tienen varias limitaciones. Primero, los valores de salida del perceptrón sólo pueden ser binarios, debido a la función de activación que se emplea. Segundo, los perceptrones sólo pueden clasificar vectores linealmente separables, es decir, los pesos determinan zonas de clasificación separadas por un hiperplano. Así, en el caso de un perceptrón con 2 entradas, se clasificarían correctamente todos los patrones sólo si se pudiera establecer una línea recta entre las clases (Méndez, y otros, 2008 pág. 219).

1.5.2 El perceptrón multicapa

El perceptrón simple es capaz de resolver problemas de clasificación e implementar funciones lógicas, como por ejemplo la función OR, pero es incapaz de implementar otras funciones simples como la función lógica XOR (Méndez, y otros, 2008 pág. 219).

Las limitaciones de las redes de una sola capa hicieron que se plantease la necesidad de implementar nuevas redes en las que se aumentase el número de capas introduciendo capas intermedias entre la capa de entrada y la capa de salida, de manera que se pudiese implementar cualquier función con el grado de precisión deseado (Méndez, y otros, 2008 pág. 219). La función que cumple dicha capa intermedia es tratar de realizar una proyección en la que resulten separables linealmente los patrones de entrada de manera que la unidad de salida pueda realizar una clasificación correcta, así el perceptrón Multicapa (Méndez, y otros, 2008 pág. 219).

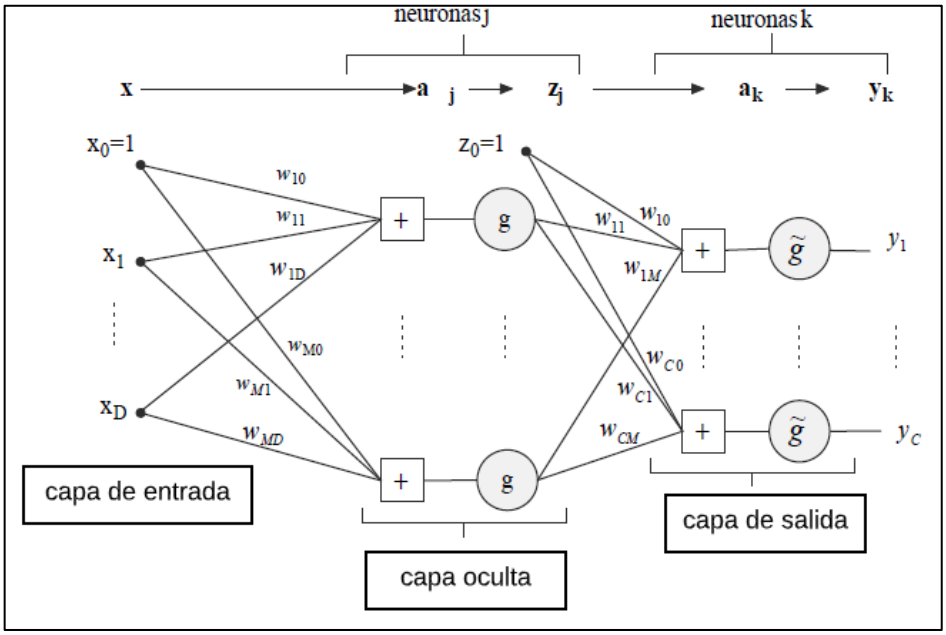


Figura 6-1 Arquitectura de un perceptrón multicapa con una capa oculta.

Fuente: (Méndez, y otros, 2008 pág. 56).

El hecho de añadir más capas sólo supone repetir las operaciones que se van a exponer, así, dicho el Perceptrón multicapa consta de D entradas, M neuronas en su capa oculta y C unidades de salida (Méndez, y otros, 2008 pág. 219). El nivel de activación aj de la neurona j de la capa oculta se calcula como una combinación lineal de las D entradas xi que recibe sobre la que, tras aplicar una función de transferencia g se obtiene la salida zj de dicha neurona:

$$z_j = g(a_j) = g\left(\sum_{i=0}^{D} w_{ji} x_i\right) \;\; ; \quad k = 1, 2, \dots, M$$

Ecuación 1-12: El perceptrón multicapa.

donde w_{ji} es el peso asociado a la neurona j y la entrada x_i. De manera similar, cada salida de la red se obtiene como una suma ponderada de las salidas de las unidades de la capa oculta, sobre la que se aplica una función de transferencia, es decir, la salida de la neurona k viene dada por:

$$y_k = \tilde{g}(a_k) = \tilde{g}\left(\sum_{j=0}^{M} w_{kj} z_j\right) = \tilde{g}\left(\sum_{j=0}^{M} w_{kj} g\left(\sum_{i=0}^{D} w_{ji} x_i\right)\right); \quad k = 1, 2, \dots, M$$

Ecuación 1-13: Función de transferencia del perceptrón multicapa.

1.6 Algoritmo de retropropagación del error

El perceptrón multicapa basa el aprendizaje de sus pesos en una regla de ajuste del error, trata de determinar los pesos de las conexiones sinápticas de manera que las salidas de la red coincidan

con las salidas deseadas, o por lo menos, sean lo más próximas posibles. Como veremos, el algoritmo de aprendizaje del perceptrón multicapa utiliza el método de descenso del gradiente para ajustar los pesos de la red (Méndez, y otros, 2008 pág. 220).

Dicho método se realiza comenzando por la capa de salida, según el error cometido, y propagando este error a las capas anteriores, hasta llegar a la capa de las unidades de entrada, de ahí que se denomine algoritmo de retropropagación del error (backpropagation algorithm) (Méndez, y otros, 2008 pág. 220).

Fases del algoritmo de retropropagación del error:

- Primera fase. - Se aplica un patrón a las entradas de la red y su efecto se propaga a través de la misma, capa a capa finalmente, la red presenta un conjunto de salidas como respuesta a dicho patrón de entrada (Méndez, y otros, 2008 pág. 220).

- Segunda fase. - Los pesos de la red se recalculan de acuerdo con una regla de ajuste del error, se calcula el valor de la función de error que compara la respuesta actual de la red y la respuesta deseada, y este error se propaga hacia atrás (Méndez, y otros, 2008 pág. 220). La función de error se expresa de la siguiente manera:

$$E^n = \frac{1}{2}\sum_{k=0}^{C}(e_k^n)^2$$

Ecuación 1-14: Algoritmo de retropropagación del error.

Donde el factor $\frac{1}{2}$, que no altera el resultado de la función, se introduce para facilitar los cálculos posteriores, y error e_k^n se define como:

$$e_k^n = y_k^n - t_k^n$$

Ecuación 1-15: Algoritmo de retropropagación de error

1.7 Métodos avanzados de aprendizaje

Para resolver problemas por medio de redes neuronales han surgido varios métodos de aprendizaje, la mayoría basados en modificaciones del algoritmo original de retropropagación (Méndez, y otros, 2008 pág. 220).

1.7.1 Métodos basados en mínimos cuadrados lineales.

Se basan en la minimización del error cuadrático medio, pero lo hacen entre la señal a que recibe la neurona de salida antes de aplicar la función de activación no lineal y una versión de la salida

deseada modificada adecuadamente. Con ello se consigue eliminar en algunos casos, y reducir en otros, las iteraciones necesarias en el entrenamiento (Méndez, y otros, 2008 pág. 220).

1.7.2 Métodos de segundo orden.

El algoritmo de retropropagación usa principalmente la información proporcionada por la primera derivada de la función de error. Sin embargo, se ha demostrado que el uso de las derivadas segundas incrementa la velocidad del aprendizaje. Los algoritmos que las usan, llamados algoritmos de segundo orden, están entre los más rápidos en términos de convergencia (Méndez, y otros, 2008 pág. 221).

1.7.3 Paso de aprendizaje adaptativo.

En el algoritmo de retropropagación la tasa de aprendizaje es constante; sin embargo, variar su valor podría acelerar y mejorar el aprendizaje. Se han desarrollado diversos métodos heurísticos para la adaptación dinámica de la tasa de aprendizaje (Méndez, y otros, 2008 pág. 221).

1.7.4 Inicialización apropiada de los pesos.

El punto de partida del entrenamiento para la red neuronal, determinado por los valores iniciales de los pesos, esto influye en la calidad de la convergencia de la red hacia el mínimo de la función de error. Por este motivo, se han propuesto diversos esquemas para la correcta inicialización de los pesos en la entrada de la red neuronal (Méndez, y otros, 2008 pág. 221).

1.7.5 Versiones por lotes y estocástica.

El algoritmo de retropropagación del error permite calcular el error y actualizar los pesos, bien sobre todo el conjunto de entrenamiento, esta opción puede acelerar el tiempo de convergencia cuando se manejan conjuntos de entrenamiento de gran tamaño en problemas de clasificación (Méndez, y otros, 2008 pág. 221).

1.8 Machine Learning

Machine Learning es una técnica de análisis de datos muy empleada actualmente que enseña a los computadores a hacer lo que resulta natural para las personas y los animales: aprender de las experiencias vividas (Méndez, y otros, 2008 pág. 221). Los algoritmos de machine learning emplean métodos de cálculo para “aprender” información directamente de los datos sin depender de una ecuación predeterminada como modelo, los algoritmos mejoran su rendimiento de forma adaptativa a medida que aumenta el número de muestras disponibles para el aprendizaje (Méndez, y otros, 2008 pág. 221).

Los algoritmos de Machine Learning encuentran patrones naturales en los datos que generan conocimiento y contribuyen a tomar mejores decisiones y a realizar mejores predicciones, se utilizan a diario para tomar decisiones cruciales en diagnósticos médicos, trading de acciones, previsión de la carga energética, etc (Méndez, y otros, 2008 pág. 221). Uno de los ejemplos más claros son los sitios multimedia confían en el aprendizaje automático para cribar millones de opciones con objeto de ofrecerle recomendaciones sobre canciones o películas, los minoristas lo utilizan para obtener información sobre el comportamiento de compra de sus clientes (Méndez, y otros, 2008 pág. 222).

El aprendizaje automático emplea dos tipos de técnicas: el aprendizaje supervisado, que entrena un modelo con datos de entrada y salida conocidos para que pueda predecir salidas futuras, y el aprendizaje no supervisado, que encuentra patrones ocultos o estructuras intrínsecas en los datos de entrada (Méndez, y otros, 2008 pág. 222).

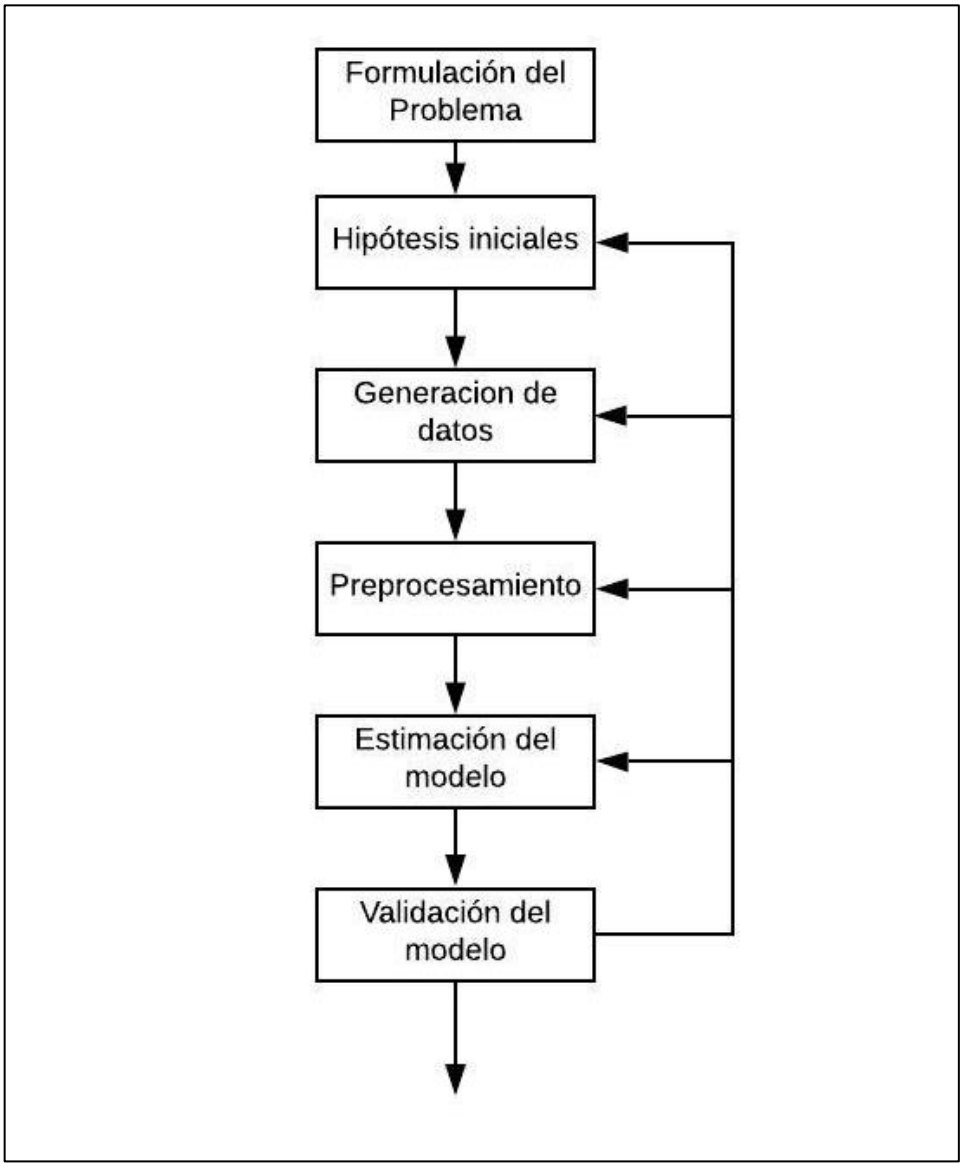


Gráfico 5-1: Esquema de procesamiento para estimar modelos de aprendizaje

Fuente: (Martinsanz, y otros, 2011 pág. 45)

1.8.1 Formulación del problema.

Todo conocimiento surge de la formulación, aunque sea implícita, de un problema, que no suele plantearse desde un desconocimiento absoluto aquello que se desea resolver, sino que parte de unas ciertas hipótesis acerca del modelo que se desee encontrar (Martinsanz, y otros, 2011 pág. 26).

1.8.2 Hipótesis Iniciales.

Se debe establecer un conjunto de datos de entrada y de salida y una dependencia de la que se tiene un conocimiento parcial, que se desea estimar a partir de experimentales, las hipótesis erróneas también sirven ya que obligan a realizar observaciones más finas (Martinsanz, y otros, 2011 pág. 26).

1.8.3 Generación de datos.

Los datos pueden ser, bien generados de forma controlada mediante la realización de experimentos o recopilados median observación del medio sin posibilidad de influir una su generación (Martinsanz, y otros, 2011 pág. 26). Los datos se dividen en dos conjuntos: unos para entrenamiento o estima del modelo más óptimo y otros para probar el modelo determinado (Martinsanz, y otros, 2011 pág. 27).

1.8.4 Procesamiento Previo.

Los datos procedentes de sensores, o de una base de datos, se tratan para ser codificados de una forma conveniente para su procesamiento; este depende del dominio especifico del problema y normalmente conlleva una reducción de la dimensión de los datos, extrayéndose un conjunto de características o propiedades que contienen la información relevante de los datos, o la que se considera más relevante, percepción que puede cambiar según se mejora en el conocimiento del modelo (Martinsanz, y otros, 2011 pág. 27).

Este proceso es conocido como extracción de características, y es un aspecto tan importante que muchas veces el éxito en algunas aplicaciones se debe a la elección adecuada de las características que se utilizan para codificar los datos (Martinsanz, y otros, 2011 pág. 27). El reconocimiento de patrones el término de extracción de características se refiere a los procedimientos para extraer números de los patrones que de alguna forma representa la información crucial contenida en dichos patrones (Martinsanz, y otros, 2011 pág. 27).

1.8.5 Estima del modelo.

Este proceso se denomina también aprendizaje, o entrenamiento y consiste en seleccionar un modelo usando los datos como guía. La selección con frecuencia necesita alguien o una medida que permite comparar cuán bien se ajusta a cada modelo a los datos o bien de una medida que permita cuantificar la fiabilidad de la predicción del modelo a nuevos datos (Martinsanz, y otros, 2011 pág. 27).

1.8.6 Validación del modelo.

En este proceso se trata de garantizar que el modelo es útil, no solo para los datos usados en la estimación, sino también para otro conjunto de datos que se pueden denominar datos de validación o datos de entrenamiento en ocasiones, el modelo puede ajustarse muy bien los datos de entrenamiento, pero puede tener muy mal las prestaciones en el test (Martinsanz, y otros, 2011 pág. 28). Se dice entonces que el modelo está sobre ajustado, uno de los problemas de aprendizaje es como ajustar la complejidad del modelo, si es muy sencillo podría no aprender lo suficiente y si es muy complejo, podría comportarse erróneamente ante nuevos datos (Martinsanz, y otros, 2011 pág. 28).

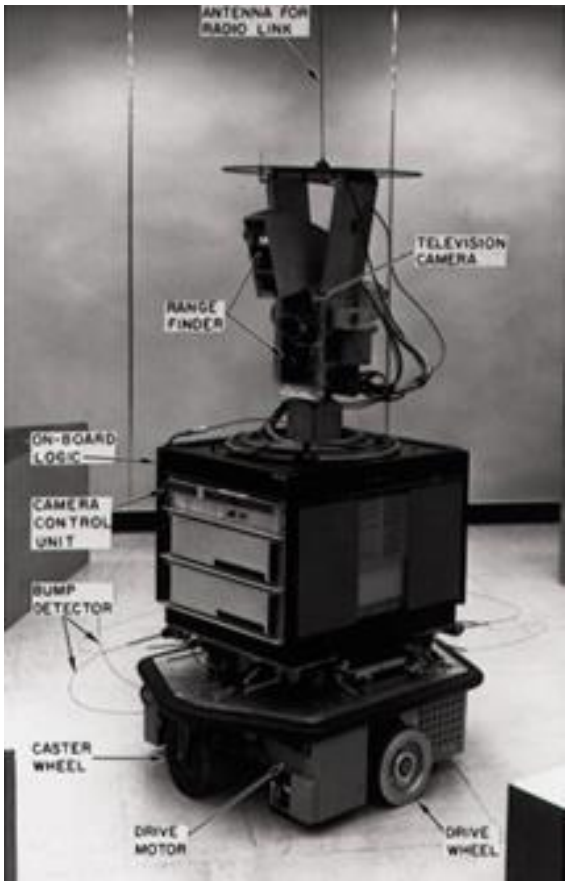


Figura 7-1 Primer robot móvil que podía tomar decisiones año 1971.

Fuente: (Martinsanz, y otros, 2011 pág. 45)

1.9 El árbol de manzano

El origen del manzano aun genera controversia. Según Tamaro (1987) es originario de las zonas templadas de Europa, de las montañas del Cáucaso y del Asia Central, se encuentra en estado silvestre y cultivado desde la prehistoria (Heras, 2019 pág. 5).

Los manzanos se adaptan a un amplio rango de condiciones climáticas, suelos diversos y sistemas de cultivo. Las plantaciones se encuentran en varias regiones de Asia y otros continentes. En Europa, desde los países escandinavos en el norte, hasta Italia en el Sur; en Sudáfrica, Australia y Nueva Zelanda; en Estados Unidos y Canadá; y en países de América del Sur (Heras, 2019 pág. 5).

El cultivo del manzano en la zona alta del Ecuador, se localiza en las provincias de Tungurahua, Chimborazo, Cotopaxi, Cañar, Azuay y Loja, entre altitudes que van de 2.650 a 3.200 msnm. Para un satisfactorio comportamiento de este frutal en la zona alta, se requiere de la presencia de un período relativamente abrigado y lluvioso, que favorezca el desarrollo vegetativo y la fructificación; y otro frio seco y ventosos con presencia de nubosidad, que permita el receso vegetativo o agostamiento; una temperatura media de 13 grados C, permite una acumulación de frio entre 400 a 600 horas (AGROPECUARIAS, 1992 pág. 2).

1.9.1 Enfermedades y su Control

En el manzano inciden varias enfermedades, que reducen su capacidad de producción, así como también la calidad de la fruta afectando directamente a los productores de esta fruta. Entre las más importantes tenemos (González, y otros, 2014 pág. 5).

1.9.1.1 Sarna del manzano.

Esta enfermedad ataca hojas, flores, frutos y ramas, los síntomas en hojas comienzan con manchas pardas circulares, que luego se vuelven verde oliváceas, de aspecto aterciopelado por la presencia de las fructificaciones asexuales del patógeno (González, y otros, 2014 pág. 5). Estas manchas inicialmente observadas en la cara donde se produjo la infección, con el tiempo se necrosan y aumentan de tamaño, interesando gran parte de la hoja inclusive se hacen visibles en la cara contraria. Las hojas afectadas se secan y caen anticipadamente (González, y otros, 2014 pág. 6).

En frutos, los síntomas pueden manifestarse en cualquier etapa del desarrollo. inicialmente aparecen pequeñas manchas de color pardo verdoso que lentamente se extienden conservando una forma redondeada, adquiriendo luego un aspecto aterciopelado (González, y otros, 2014 pág. 7).

En frutos jóvenes en crecimiento, se observan: atrofias, malformaciones, rajaduras y algunas veces caída de los mismos. En ataques a frutos próximos a su cosecha, las lesiones son menos

marcadas y generalmente consisten en un reticulado suberificado de la epidermis (González, y otros, 2014 pág. 8).

Figura 8-1: Sarna del manzano

Fuente: (Diego Gordón, 2020).

1.9.1.2 Minadora en círculos.

De las crisálidas o capullos que han invernado en hojas caídas o rugosidades de la corteza salen, en primavera, pequeñas mariposillas que depositan sus huevos en las hojas del manzano. Nacen las orugas, que penetran en el interior de la hoja donde excavan minas o galerías concéntricas; estas, al principio aparecen como una manchita parduzca y redondeada, para continuar ensanchándose regularmente hasta alcanzar un diámetro de 7-10 mm (ORAIN, 2017 pág. 8).

Las orugas abandonan las galerías y tejen un capullo blanco y alargado cubierto por una maraña de hilos finos que dan origen a la siguiente generación. La crisalidación puede realizarse sobre las hojas, en las grietas de la corteza de los árboles o en la cavidad peduncular de los frutos. Habitualmente hay tres o cuatro generaciones al año (ORAIN, 2017 pág. 8).

Los daños serían importantes si el número de galerías fuese excesivo, lo cual alteraría la actividad de las hojas. Se recomiendan como medios de control lo indicado para la minadora sinuosa (ORAIN, 2017 pág. 9).

Figura 9-1: Minadora en círculos.

Realizado por: Diego Gordón, 2020.

1.9.1.3 Líquenes.

Se denomina liquen a una asociación entre alga y hongo de carácter simbiótico, es decir que tanto hongo como alga obtienen beneficios de la misma, el alga surte de alimentos al hongo y éste proporciona al alga humedad y protección contra las altas intensidades de luz (ORAIN, 2017 pág. 9).

Sin ser parásitos estrictos de los manzanos sí les causan daños indirectos pues sirven de cobijo a diversas plagas como ácaros e insectos, y por otra parte al retener mucha humedad puede producirse la desorganización de ciertos tejidos especialmente si existen heridas (ORAIN, 2017 pág. 10).

Los medios que se aconsejan para combatir los líquenes son culturales como labores profundas, drenaje del terreno, podas equilibradas que favorezcan una buena aireación,etc (ORAIN, 2017 pág. 9).

Figura 10-1: Líquenes.

Fuente: (Diego Gordón, 2020).

.

1.10 Matlab

Millones de ingenieros y científicos de todo el mundo usan MATLAB® para analizar y diseñar los sistemas y productos que transforman nuestro mundo. MATLAB está presente en sistemas de seguridad activa de automóviles, naves espaciales interplanetarias, dispositivos de monitorización de la salud, redes eléctricas inteligentes y redes móviles LTE. Se utiliza para aprendizaje automático, procesamiento de señales, procesamiento de imágenes, visión artificial, comunicaciones, finanzas computacionales, diseño de control, robótica y muchos otros campos (MathWorks, 1994-2020 pág. 4).

La interacción se realiza mediante instrucciones (denominadas comandos), y también mediante funciones y programas en un lenguaje estructurado (Cuenca, y otros, 2014). Los objetos básicos con los cuales opera MATLAB son matrices, la asignación de memoria a cada variable la realiza MATLAB en forma dinámica y eficiente, por lo que no son necesarias las declaraciones de variables antes de su uso (Cuenca, y otros, 2014 pág. 36).

Las características principales de Matlab:

- Lenguaje de alto nivel para cálculos científicos y de ingeniería.
- Entorno de escritorio optimizado para la exploración iterativa, el diseño y la solución de problemas.
- Gráficas para visualizar datos y herramientas para crear diagramas personalizados.
- Aplicaciones para ajustar curvas, clasificar datos, analizar señales, ajustar sistemas de control y muchas otras tareas.
- Toolboxes complementarias para una amplia variedad de aplicaciones científicas y de ingeniera (MathWorks, 1994-2020 pág. 5).

La profusión de cámaras ha generado cantidades gigantescas de datos de imagen y vídeo, por lo que ha aumentado la demanda relacionada con el procesamiento de imágenes para descifrar y comprender tales datos (MathWorks, 1994-2020 pág. 5).

Mediante el uso de MATLAB, ingenieros y otros expertos han desplegado miles de aplicaciones de machine learning, MATLAB simplifica las partes más difíciles y complicadas del machine learning y todo esto lo logra gracias a (MathWorks, 1994-2020 pág. 5):

- Apps de apuntar y hacer clic para entrenar y comparar modelos.
- Técnicas avanzadas de procesamiento de señales y extracción de funcionalidades.
- Ajuste automático de hiperparámetros y selección de funcionalidades para optimizar el rendimiento de los modelos.

- Capacidad de usar el mismo código para escalar el procesamiento para big data y clusters.
- Generación automatizada de código C/C++ para aplicaciones embebidas y de alto rendimiento.
- Algoritmos de clasificación, regresión y clustering de uso común para el aprendizaje supervisado y no supervisado.
- Ejecución más rápida que con código abierto en la mayor parte de los cálculos estadísticos y de machine learning (MathWorks, 1994-2020 pág. 6).

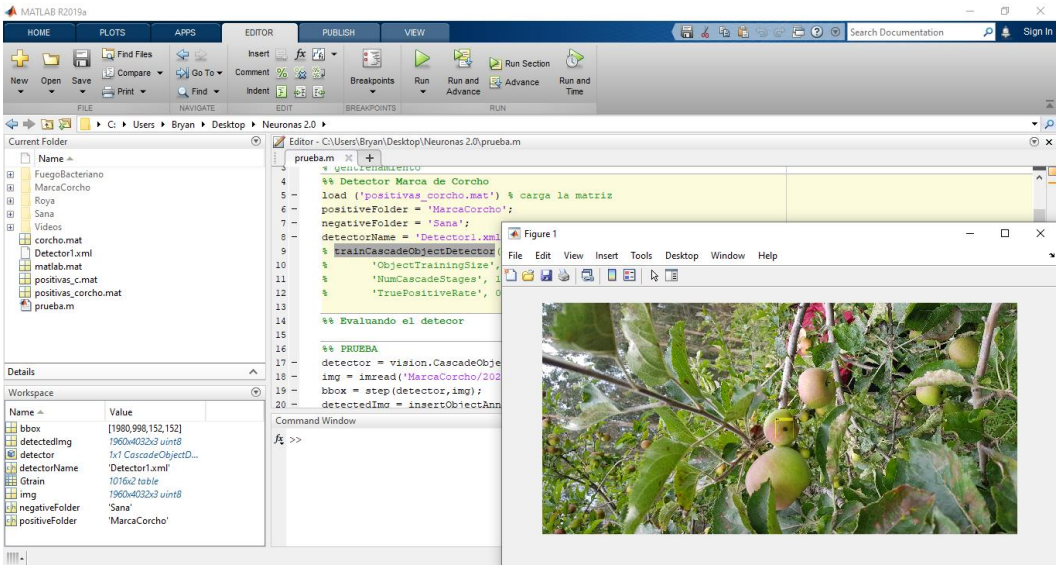


Figura.11-1: Entorno de programación de Matlab.

Fuente: (Diego Gordón, 2020).

Detección de objetos con Matlab

En la detección de objetos con Matlab se analizan las distintas imágenes para la extracción de características, y mediante la construcción de vectores tomados de una base de datos se realiza el aprendizaje automático y así dar forma al clasificador, todos los pasos que se indican en la siguiente figura permiten crear el sistema de reconocimiento y detección de objetos (Ramos, 2016 pág. 26).

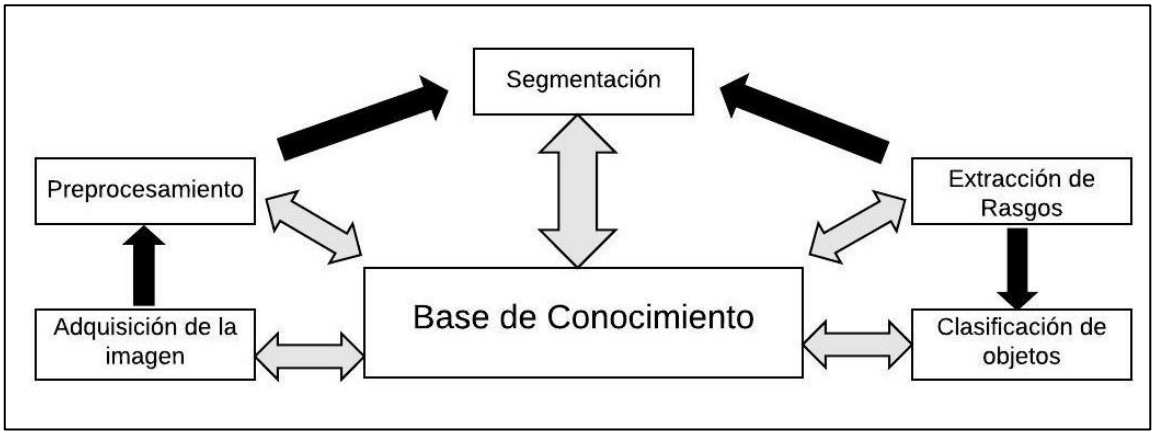


Gráfico 6-1: Procesos básicos en el reconocimiento de objetos.

Fuente: (Diego Gordón, 2020).

Donde:

1. **Adquisición de la imagen. –** Se basa en capturar la escena del mundo real para su procesamiento, almacenamiento y transmisión.
2. **Preprocesamiento. –** En este proceso se aplican técnicas de mejoramiento de contraste, reducción de ruido, etc., de modo que la imagen se adecue a las siguientes etapas.
3. **Segmentación.** – Seleccionar los objetos de interés de la imagen.
4. **Extracción de rasgos.** – Se debe describir numéricamente la naturaleza de los objetos segmentados como su forma, color, textura, etc.
5. **Clasificación.** – Asignar una clase, categoría a cada objeto segmentado de la imagen basado en sus rasgos.

Para que exista el reconocimiento de un objeto, primeramente, se debe realizar su correcta detección, al analizar una imagen cualquiera el sistema no sabe que objeto se va a reconocer, por lo que se debe realizar previamente un entrenamiento del clasificador.

1.10.1 Simulación de Sistemas de Visión Artificial en MATLAB

La Visión Artificial tiene como objetivo generar descripciones inteligentes y útiles de escenas y secuencias visuales, así como de los objetos que aparecen en ellas, mediante la realización de operaciones sobre imágenes y vídeos. (MathWorks, 2018 pág. 2)

Entre las tareas que podemos mencionar de visión artificial están:

- Reconstrucción de escenas
- Detección, reconocimiento, identificación de objetos.
- Restauración de imágenes
- Análisis de movimientos.

1.10.2 Detector de objetos en cascada

Matlab tiene clasificadores previamente entrenados para detectar caras frontales, caras de perfil, narices, ojos y la parte superior del cuerpo. Sin embargo, estos clasificadores no siempre son suficientes para una aplicación en particular. Computer Vision Toolbox, proporciona la trainCascadeObjectDetectorfunción de entrenar un clasificador personalizado. (MathWorks, 2018 pág. 3)

El detector de objetos en cascada Computer Vision Toolbox puede detectar categorías de objetos cuya relación de aspecto no varía significativamente entre los objetos de interés, el detector usa un clasificador en cascada para decidir si la ventana contiene el objeto de interés. El tamaño de la ventana varía para detectar objetos a diferentes escalas, pero su relación de aspecto permanece fija. El detector es muy sensible a la rotación fuera del plano, porque la relación de aspecto cambia

para la mayoría de los objetos en 3D. Por lo tanto, debe entrenar un detector para cada orientación del objeto. No funcionará entrenar un solo detector para manejar todas las orientaciones. (MathWorks, 2018 pág. 3)

Elija la función que se adapte al tipo de detección de objetos que necesita. El trainCascadeObjectDetectoradmite tres tipos de características: Haar, patrones binarios locales (LBP) e histogramas de gradientes orientados (HOG). Las funciones Haar y LBP se utilizan a menudo para detectar rostros porque funcionan bien para representar texturas de escala fina. Las funciones HOG se utilizan a menudo para detectar objetos como personas y automóviles. (MathWorks, 2018 pág. 3)

1.10.2.1 Función HOG

HOG (Histogram of Oriented Gradients) se basan en la orientación del gradiente en áreas locales de una imagen, el descriptor HOG permite aprovechar de forma eficiente la información del gradiente a partir de combinar esta información en forma de histogramas orientados locales, que se calculan en celdas de pequeño tamaño, las cuales se distribuyen de forma uniforme por toda la imagen estos histogramas nos proporcionan información de las orientaciones de los contornos que dominan en cada una de las posiciones de la imagen, esta información nos va a permitir distinguir la forma de los objetos presentes en una imagen y es una buena base para detectar y reconocer dichos objetos gracias a esa información, podemos ver la frontera entre un objeto y otro. (Ramos, 2016 pág. 48)

1.10.2.2 Función Haar

Haar, realizan una codificación de diferencia de intensidades en la imagen, generando características de contornos, puntos y líneas, mediante la captura de contraste entre regiones, la extracción de características es realizada aplicando a las imágenes filtros con bases Haar estos filtros pueden ser calculados eficientemente sobre la imagen integral, son selectivos en la orientación espacial y frecuencia, y permiten ser modificados en escala y orientación. (GUEVARA, y otros, 2008 pág. 35)

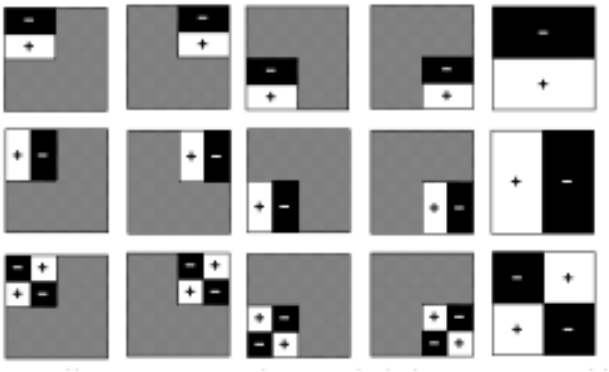


Figura.12-1: Visualización de una secuencia Haar

Fuente: (GUEVARA, y otros, 2008 pág. 35)

1.10.2.3 Función LBP

Local Binary Pattern (LBP), es un descriptor de textura que se caracteriza por su robustez ante cambios de iluminación. El proceso para obtener dicho descriptor es transformar la imagen a escala de grises, luego se toma información de todos los vecinos de cada pixel, comparando el pixel central con los de su alrededor y en base a un umbral se obtiene un número binario. (SOLANO AGUILAR, 2018)

1.11 Internet de las cosas (IoT)

"Internet of things" se define como la interconexión y la interacción de lo digital y el mundo físico, en el que la tecnología permite integrar "cosas" físicas a las redes de información a través de infraestructuras de Internet existentes y emergentes. Es decir, IoT es una plataforma para conectar personas, objetos, y entornos para informar y permitir la visibilidad, compromiso, y la innovación (Inesdi, y otros, 2015 pág. 3).

En el último año unos 4.000 millones de dispositivos se han conectado a Internet según estudios realizados por la empresa Cisco (Inesdi, y otros, 2015). Por otro lado, la compañía Intel ha predicho que la cifra podría llegar a alcanzar los 200.000 millones en menos de cinco años y que para el 2020 habrá aproximadamente 26 objetos inteligentes conectados por humano (Inesdi, y otros, 2015 pág. 4).

Las interacciones entre personas, procesos, cosas y datos están creando nuevos tipos de aplicaciones y servicios inteligentes donde intervienen LOS SENSORES y el tipo de CONECTIVIDAD, y donde éstos juegan un papel fundamental. (Incipy, 2015 pág. 6)

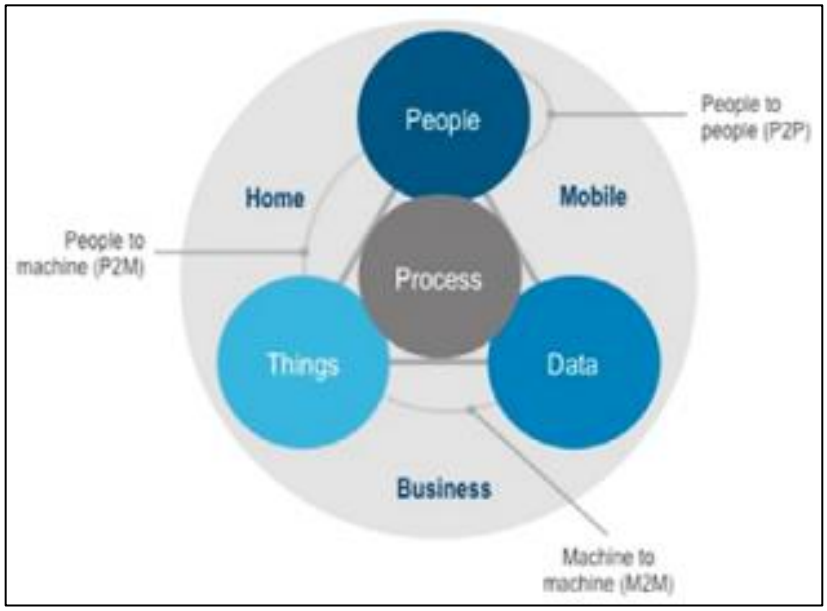


Figura.13-1 Transformación Digital por la aplicación de IoT**.**

Fuente: (Incipy, 2015 pág. 15)

1.12 VANT

VANT (Vehículo Aéreo No Tripulado) de uso civil es una aeronave a propulsión, no tripulada y reutilizable que opera mediante control a distancia y autónomamente, el uso de esta tecnología, no tiene límites y plantea muchos interrogantes, la adopción de nuevas tecnologías que permiten ser más eficientes, es lo que se impone, por ello, es importante conocer las características principales de este tipo de Tecnología. (Addati, y otros, 2014 pág. 2)

Algunas de las áreas de aplicación más frecuentes son:

- Imágenes y video aéreo.
- Monitoreo y vigilancia.
- Búsqueda y rescate.
- Mapeo de terrenos.
- Gestión de emergencias.

Figura.14-1 Vehículo aéreo no tripulado VANT

Fuente: (Diego Gordón, 2020).

CAPITULO II

2 MARCO METODOLÓGICO

En el presente capitulo, se detalla la metodología que se usa para el uso de algoritmos basados en machine learning para la detección de enfermedades en plantas frutales de piso climático templado utilizando procesamiento de imágenes caso práctico manzano. Para lograr este objetivo se va a ser uso de la herramienta Matlab, el cual ofrece toolboxes especializadas para trabajar con machine learning, redes neuronales, deep learning, visión artificial.

2.1 Metodología de la investigación

Para el desarrollo del presente trabajo se aplican una serie de técnicas, métodos y conocimientos con el propósito de recabar, clasificar y analizar correctamente todos los datos obtenidos. Las técnicas y procedimientos se detallarán a continuación.

2.2 Tipo de Investigación

El presente trabajo de investigación se clasifica en aplicativo y experimental.

2.2.1 Aplicativa

La Investigación Aplicada tiene por objetivo principal resolver un determinado problema o planteamiento, enfocándose en la búsqueda del conocimiento para su aplicación y, por ende, para el enriquecimiento del desarrollo cultural y científico.

2.2.2 Experimental

Ya que se basa en una serie de pruebas realizadas en escenarios de laboratorio controlados, en las que se observa los elementos más importantes del tema de estudio que se investiga para obtener una captación de los fenómenos a primera vista.

2.3 Métodos y Técnicas

Los métodos y técnicas utilizados para la presente investigación son los siguientes:

2.3.1 Métodos

Para el presente trabajo de investigación se utilizará el Método de Análisis el cual permitirá la identificación de cada una de las partes, las cuales caracterizan la realidad del objeto a estudio para que de esta manera se establezca una relación causa-efecto entre los elementos que componen el mismo.

De igual manera se hará uso del Método Científico que es el proceso mediante el cual una teoría científica es validada o bien descartada, esta contiene una serie ordenada de pasos a seguir para la resolución de un problema determinado, cuyas etapas son:

- Observación
- Planteo de un problema

- Recopilación de datos
- Formulación de hipótesis
- Experimentación
- Conclusión.

2.3.2 Técnicas

Las técnicas a utilizarse en el presente trabajo son:

2.3.2.1 Búsqueda de información

Esta técnica permite obtener la información necesaria acerca del objeto a estudio de la investigación y para su posterior desarrollo, utilizando fuentes secundarias disponibles.

2.3.2.2 Pruebas

Permite realizar experimentos en escenarios de laboratorio controlados.

2.3.2.3 Observación

La técnica de observación permite determinar resultados de las pruebas realizadas en los escenarios de laboratorio.

2.3.2.4 Análisis

La técnica de análisis me permite determinar los resultados de la investigación realizada.

2.4 Instrumentos

Los instrumentos usados para recopilar los datos son los siguientes:

2.4.1 Matlab

Mediante el uso de MATLAB, ingenieros y otros expertos han desplegado miles de aplicaciones de machine learning. MATLAB simplifica las partes más difíciles del machine learning (https://es.mathworks.com/solutions/machine-learning.html).

2.4.2 GadgetsPack

Se trata de un software que ha sido desarrollado para entornos Windows, concretamente para Windows 7, 8 y 10, y que cuenta con una versión gratuita para que podamos ver desde la barra de tareas el consumo de CPU y RAM, así como del almacenamiento o los datos de red, de un sólo vistazo (https://entropy6.com/xmeters/).

2.4.3 Google Heard Pro.

El mapa de Google Earth está compuesto por una superposición de imágenes obtenidas por imágenes satelitales, información geográfica proveniente de modelos de datos SIG de todo el mundo y modelos creados por computadora. El programa está disponible en varias licencias, pero la versión gratuita es la más popular, disponible para dispositivos móviles, tabletas y computadoras personales (https://www.google.com/intl/es/earth/).

2.4.4 ThingSpeak.

Es un servicio de plataforma de análisis de IoT que le permite agregar, visualizar y analizar flujos de datos en vivo en la nube. Puede enviar datos a ThingSpeak desde sus dispositivos, crear una visualización instantánea de datos en vivo y enviar alertas (https://thingspeak.com/).

Analice y visualice sus datos con MATLAB, el almacenamiento de datos en la nube proporciona un fácil acceso a sus datos, usando herramientas analíticas en línea, puede explorar y visualizar datos además de descubrir relaciones, patrones y tendencias en los datos (https://thingspeak.com/).

2.4.5 Sistema Android.

Android es el nombre de un sistema operativo que se emplea en dispositivos móviles, este sistema operativo puede adaptarse a múltiples resoluciones de pantalla y soporta conexiones WiFi, Bluetooth, LTE, CDMA, GSM/EDGE, HSPA+ y UMTS, entre otras. También permite el envío de mensajes MMS y SMS, cuenta con navegador web, posibilita el desarrollo de streaming y está capacitado para trabajar con archivos MP3, GIF, PNG, WAV, MIDI y otros formatos multimedia (https://www.android.com/).

2.4.6 Blogger

Un Blogger se puede definir como una persona o un conjunto de personas que administra un sitio o red social en internet con el objetivo de entretener, informar o vender. Es el Blogger quien mantiene en contacto directo con el público y se comunica con sus visitantes directamente, un Blogger puede dedicarse a publicar contenidos interesantes para el público específico de su blog, además de promover productos o servicios. (Martínez, y otros, 2010 pág. 29)

2.5 Identificación de las muestras

2.5.1 Población y Muestra

Para la verificación del adecuado nivel de precisión en el reconocimiento de enfermedades en los manzanos, las plantas de manzano se las seleccionaron como población y como muestra una serie de fotografías.

El huerto de manzano que se tomó como población del proyecto está ubicado en el cantón Mocha (-1.379677, -78.630126), el huerto tiene una superficie de 8310.59 m^2 con una población de 900 manzanos. El huerto seleccionado para el estudio tiene tres tipos de manzano coexistiendo en el mismo entorno los cuales están divididos de la siguiente manera.

Tabla 1-2: Población y Muestra del proyecto.

Especie		Población de Manzano
Nombre Común	**Nombre Científico**	
Emilia	*Manzano esperiega*	3116
Delicia	*Golden Delicious*	1715
Chilena	*McIntosh roja*	2094

Realizado por: Diego Gordón, 2020.

Para determinar el número adecuado de muestras de cada población de manzanos se utilizó la fórmula de cálculo de numero de muestras:

$$n = \frac{Z^2 * p * q * N}{e^2(N-1) + Z^2 * p * q}$$

Ecuación 2-1: Número de muestras

Donde:

n= Tamaño de la muestra

N=Poblacion y universo

Z=Nivel de confianza (97.5%)

p=probabilidad a favor

q=probabilidad en contra

e=error muestral

$$n = \frac{2.24^2 * 0.5 * 0.5 * 3116}{0.025^2(3116-1) + 2.24^2 * 0.5 * 0.5}$$

$$n = 1220.98 = 1221\ muestras\ \ de\ Manzano\ esperiega$$

$$n = \frac{2.24^2 * 0.5 * 0.5 * 1715}{0.025^2(1715-1) + 2.24^2 * 0.5 * 0.5}$$

$$n = 925.02 = 925\ muestras\ \ de\ Golden\ delicious$$

$$n = \frac{2.24^2 * 0.5 * 0.5 * 2094}{0.025^2(2094-1) + 2.24^2 * 0.5 * 0.5}$$

$$n = 1025.04 = 1025\ muestras\ \ de\ McIntosh\ roja$$

Tabla 2-2: Población y Muestra del proyecto.

Especie		**Población de Manzano**	**Tamaño de la muestra**
Nombre Común	**Nombre Científico**		
Emilia	*Manzano esperiega*	3116	1221
Delicia	*Golden delicious*	1715	925
Chilena	*McIntosh roja*	2094	1025
Total		6925	3171

Fuente: (Diego Gordón, 2020).

A la población de manzanos se les realizo una serie de tomas mediante un VANT, a los videos obtenidos se les realizo un tratamiento en Matlab que permitió obtener el número de muestra, en este caso imágenes precalculadas anteriormente.

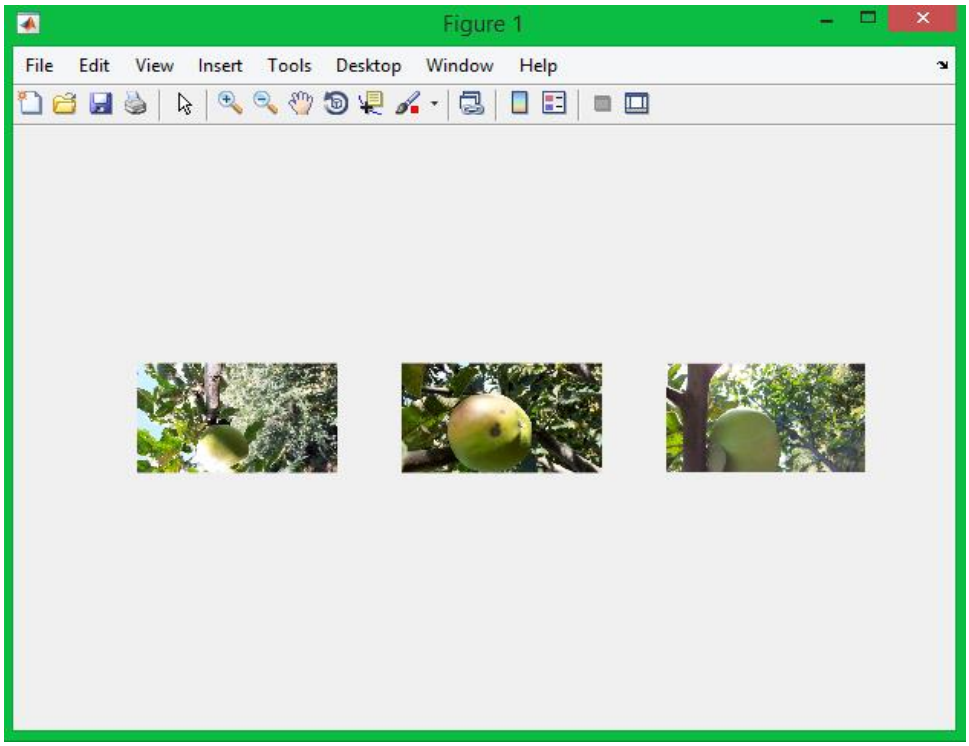


Figura 1-2: Obtención de imágenes a partir de un video obtenido por el Vant.

Fuente: (Diego Gordón, 2020).

2.6 Planteamiento de Hipótesis

La selección adecuada del algoritmo de reconocimiento de objetos en imágenes digitales permitirá determinar el mejor grado de precisión y mínimo consumo de recursos en computadoras.

Tabla 3-2: Operación conceptual de la Hipótesis

Variable	Tipo	Concepto
Algoritmo de reconocimiento de Objetos	Simple Cualitativa Independiente	Es el conjunto de algoritmos (de aprendizaje supervisado y no supervisado) existentes para el procesamiento de imágenes digitales.
Grado de precisión en la detección de objetos y consumo de recursos del computador	Compleja Cuantitativa Dependiente	Porcentaje de aciertos y fallos en el reconocimiento de enfermedades en los manzanos, y el consumo de memoria RAM y procesamiento en computadoras.

Fuente: (Diego Gordón, 2020).

Tabla 4-2: Operacionalización Metodológica de la Hipótesis de Investigación

Variable	**Indicador**	**Técnica**	**Instrumento/Fuente**
Algoritmo de reconocimiento de Objetos	Tipos de algoritmos de capaces de detectar enfermedades en manzanos.	Revisión Bibliográfica, Pruebas.	Algoritmos de reconocimiento de objetos.
Grado de precisión en la detección de objetos y consumo de recursos del computador	- % de Precisión. - Tiempo de procesamiento. - Consumo de RAM.	Analizadores de rendimiento, pruebas	Software de análisis de recursos computacionales Xmeters

Fuente: (Diego Gordón, 2020).

2.7 Desarrollo de la Aplicación.

Una vez que las herramientas para el desarrollo del proyecto fueron cuidadosamente seleccionadas, se procede a desarrollar la aplicación para detección de enfermedades de manzanos aplicando algoritmos machine learning, en la figura.14 se muestra las fases involucradas en el desarrollo de esta aplicación.

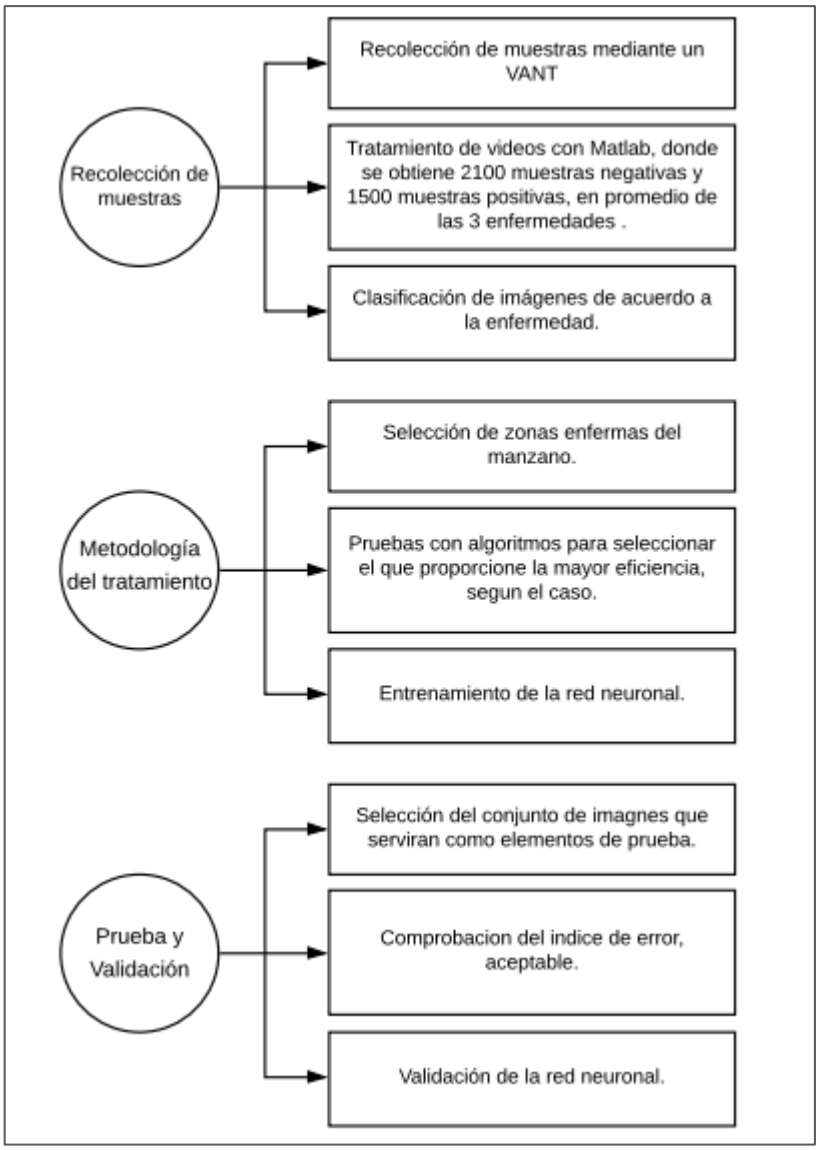


Gráfico 1-2: Fases para el reconocimiento de patrones

Fuente: (Diego Gordón, 2020).

2.7.1 Recolección de muestras

A continuación, se detalla cada una de las fases definidas en el pre procesamiento y tratamiento de las imágenes tomadas.

```
clc, clear all, close all
save('positivas_corcho','Gentrenamiento_corcho') para guardar el gentrenamiento
trainingImageLabeler para crear las Gtrain y Gtest
% Extraccion de imagenes de videos
video = 'manzana.mp4';    % Nombre del video
carpeta = 'IMG';  % Nombre de la carpeta para almacenar las imágenes
NumeroImagenes = 20000; %Número de imágenes a extraer

videoobj = VideoReader(video);
intervalo = round(linspace(1,videoobj.NumberOfFrames,NumeroImagenes));
for i=intervalo
    frame = read(videoobj,i);
    namefile = [carpeta, '\ima', num2str(i), '.jpg'];
    imwrite(frame,namefile)
end
```

Figura 2-2: Código de procesamiento de video, para extraer imágenes de Matlab.

Fuente: (Diego Gordón, 2020).

En la figura 15 se está mostrando el código con el cual se extrae imágenes de los videos tomados por el VANT, el número de muestras a extraer es de 7000 imágenes las cuales serán almacenadas en la carpeta llamada ´IMG´, además se usa un siclo for para extraer las imágenes las cuales se guardan en formato jpg, manteniendo la misma calidad que el video.

Figura 3-2: Clasificación de imágenes extraídas por el video.

Fuente: (Diego Gordón, 2020).

La clasificación de las imágenes se las realizo manualmente debido a la naturaleza variable de las enfermedades estudiadas, siendo identificadas 3 tipos de enfermedades representativas en toda la población seleccionada para el estudio.

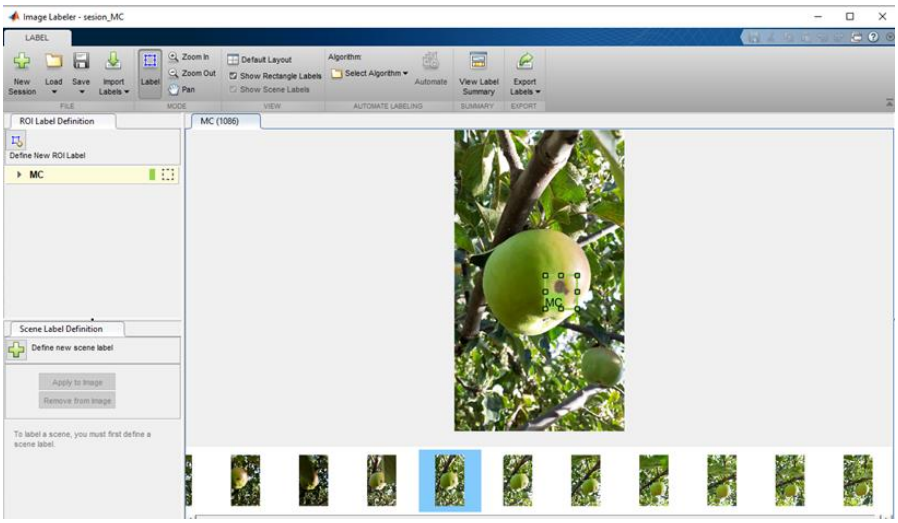


Figura 4-2: Imagen Labeler seleccionando el de interés del manzano.

Fuente: (Diego Gordón, 2020).

2.7.2 Entrenamiento de la aplicación

El entrenamiento del clasificador en cascada requiere un gran conjunto de muestras positivas y un conjunto de imágenes negativas. Debe proporcionar un conjunto de imágenes positivas con regiones de interés especificadas para usar como muestras positivas. Puede usar *Image Labeler* para etiquetar objetos de interés con cuadros delimitadores como se muestra en la *Figura16*. *Image Labeler* genera una tabla para usar en muestras positivas. También debe proporcionar un conjunto de imágenes negativas a partir de las cuales la función genera muestras negativas automáticamente. Para lograr una precisión aceptable del detector, se debe establecer un número de etapas, el tipo de característica y otros parámetros de función.

Figura 5-2: Clasificación de imágenes extraídas por el video.

Fuente: (Diego Gordón, 2020).

2.7.2.1 Selección de las zonas afectadas por las enfermedades

Para la selección de las zonas afectadas por la enfermedad en el árbol de manzano se utiliza la herramienta de Matlab *trainingImageLabeler*(etiquetadora de imágenes*)*, el proceso de etiquetado

se lo debe realizar de manera individual para cada imagen y para cada enfermedad tal como se muestra en la Figura 17.

Se debe seleccionar la función que se adapte al tipo de detección de objetos que se necesite. *TrainCascadeObjectDetector* admite tres tipos de funciones: Haar, patrones binarios locales (LBP) e histogramas de gradientes orientados (HOG). Las funciones de Haar y LBP a menudo se usan para detectar caras porque funcionan bien para representar texturas de escala fina. Las funciones HOG a menudo se utilizan para detectar objetos. Son útiles para capturar la forma general de un objeto.

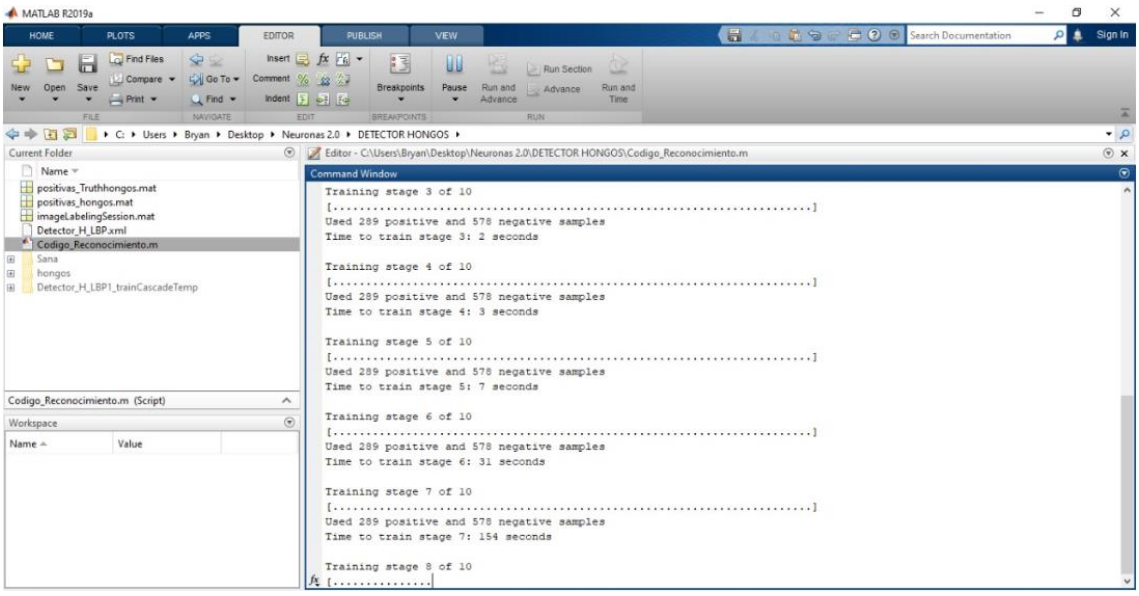


Figura 6-2: Etapas de un entrenamiento para ajustar los mejores parámetros

Fuente: (Diego Gordón, 2020).

Para la etapa de entrenamiento se probó las 3 funciones: LBP, HOG, HAAR, para determinar cuál es el la que se acopla de mejor manera a nuestro proyecto primero se probó con la función HOG con 10000 etapas en las cuales se tardó 9 horas y 43 minutos en el entrenamiento y se obtuvo resultados aceptables pero no con el índice de incidencia que necesitaba nuestro detector, una vez realizada esta prueba pasamos a realizar pruebas con la función HAAR de igual manera con 10000 etapas el tiempo estimado por Matlab para este entrenamiento fue de 27 minutos y 7 segundos por cada etapa, esto con lleva a que el tiempo estimado de entrenamiento es de 192 días por lo que se desestima esta función por el tiempo que emplea su procesamiento, al final se probó con la función LBP al igual que con las otras funciones con 10000 etapas de entrenamiento, en las cuales se tardó 19 horas y 44 minutos en el entrenamiento y con la cual se consiguió los mejores resultados en la detección de enfermedades de las plantas de manzano.

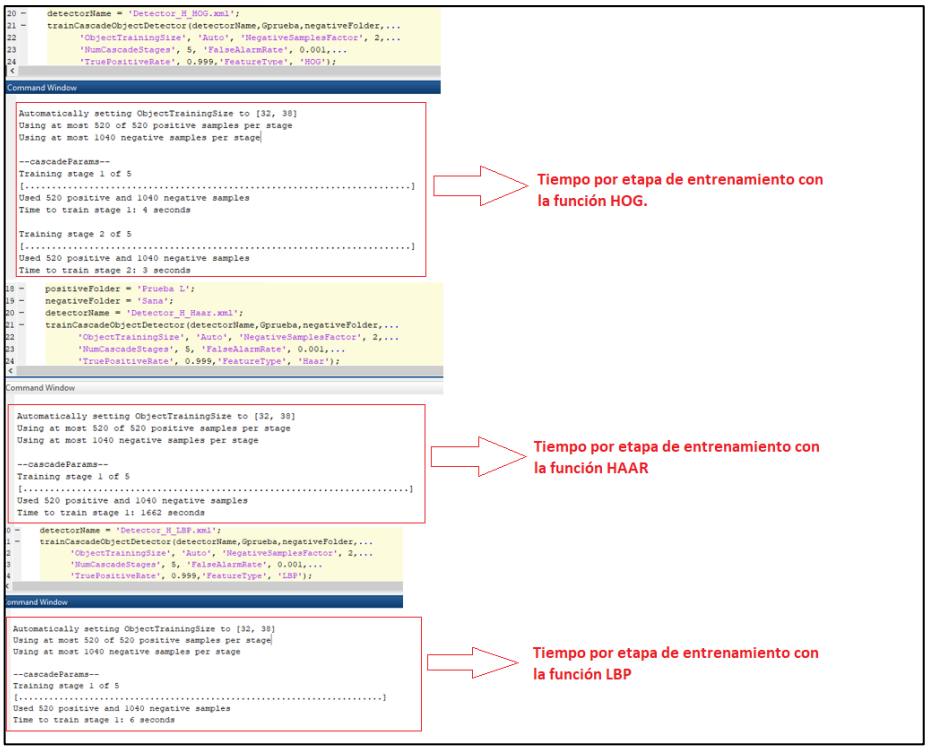


Figura 7-2: Tiempos de entrenamiento de las funciones HOG, HAAR, LBP.

Fuente: (Diego Gordón, 2020).

2.7.2.2 Creación del detector con la matriz de positivos

Se debe ejecutar la función *trainCascadeObjectDetector* varias veces para ajustar los parámetros. Para ahorrar tiempo, puede usar las funciones LBP o HOG en un pequeño subconjunto de sus datos, ya que entrenar un detector con las funciones de Haar lleva mucho más tiempo de procesamiento.

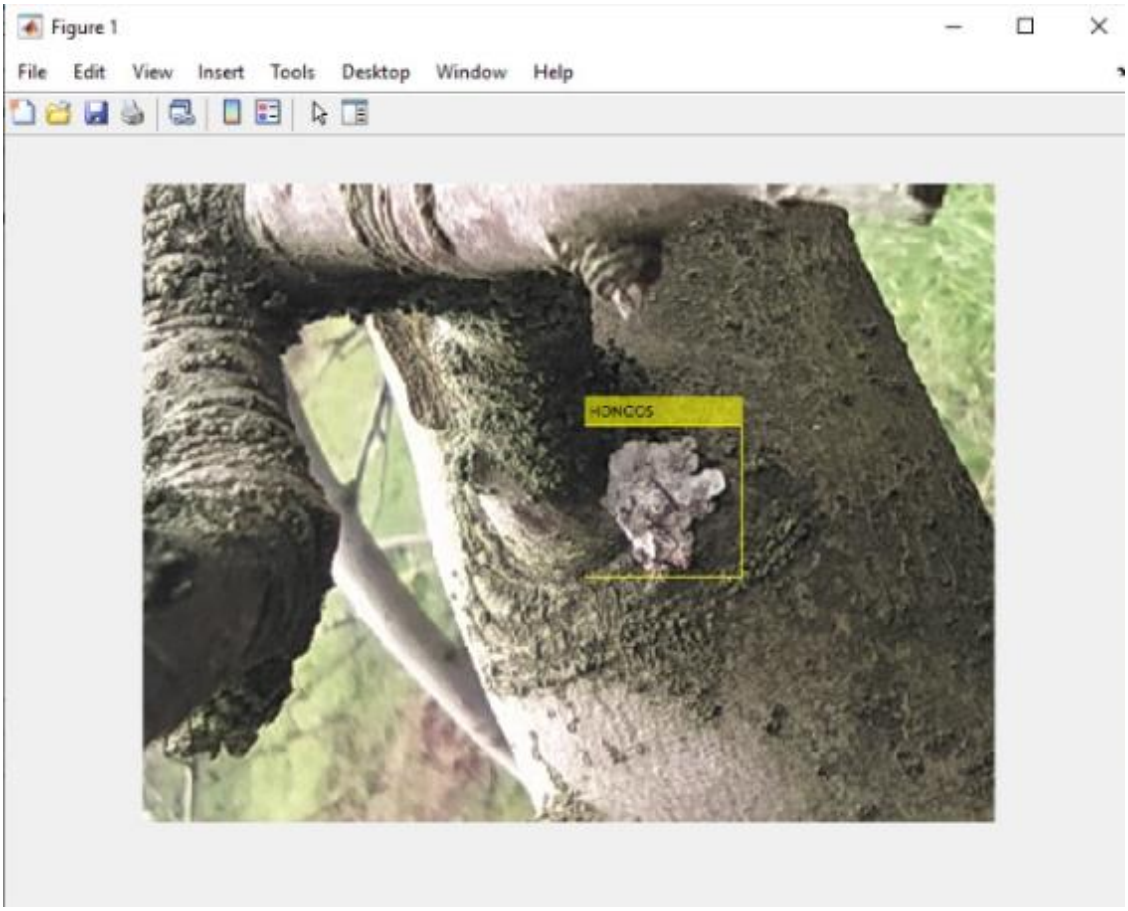


Figura 8-2: Detección del objeto a interés.

Fuente: (Diego Gordón, 2020).

El detector utiliza un clasificador en cascada para decidir si la ventana contiene el objeto de interés. El tamaño de la ventana puede variar para detectar objetos a diferentes escalas, pero su relación de aspecto permanece fija. El detector es muy sensible a la rotación fuera del plano, porque la relación de aspecto cambia para la mayoría de los objetos tridimensionales.

Cada etapa del clasificador etiqueta la región definida por la ubicación actual de la ventana deslizante como positiva o negativa. Positivo indica que se encontró un objeto de interés y negativo indica que no se encontraron objetos. Si la etiqueta es negativa, la clasificación de esta región está completa y el detector desliza la ventana a la siguiente ubicación. Si la etiqueta resulta ser un objeto positivo, el clasificador pasa la región a la siguiente etapa. El detector informa de un objeto encontrado en la ubicación actual de la ventana cuando la etapa final clasifica la región como positiva.

Las etapas están diseñadas para rechazar muestras objetos negativos lo más rápido posible. La suposición es que la gran mayoría de las ventanas no contienen el objeto de interés, pero, por el contrario, los verdaderos objetos positivos son raros y vale la pena tomarse el tiempo para verificarlos de una manera correcta.

Casos a tomar en cuenta con la clasificación de los objetos a interés:

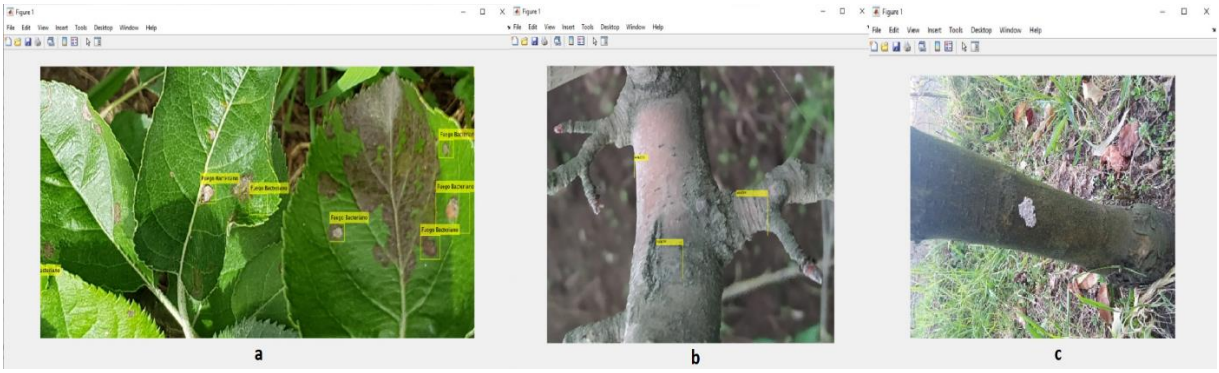


Figura 9-2: Casos a tomar encuentra a la hora de la clasificación de objetos.

Fuente: (Diego Gordón, 2020).

Donde:

a. Un verdadero positivo ocurre cuando una muestra positiva se clasifica correctamente.
b. Un falso positivo ocurre cuando una muestra negativa se clasifica erróneamente como positiva.
c. Se produce un falso negativo cuando una muestra positiva se clasifica erróneamente como negativa.

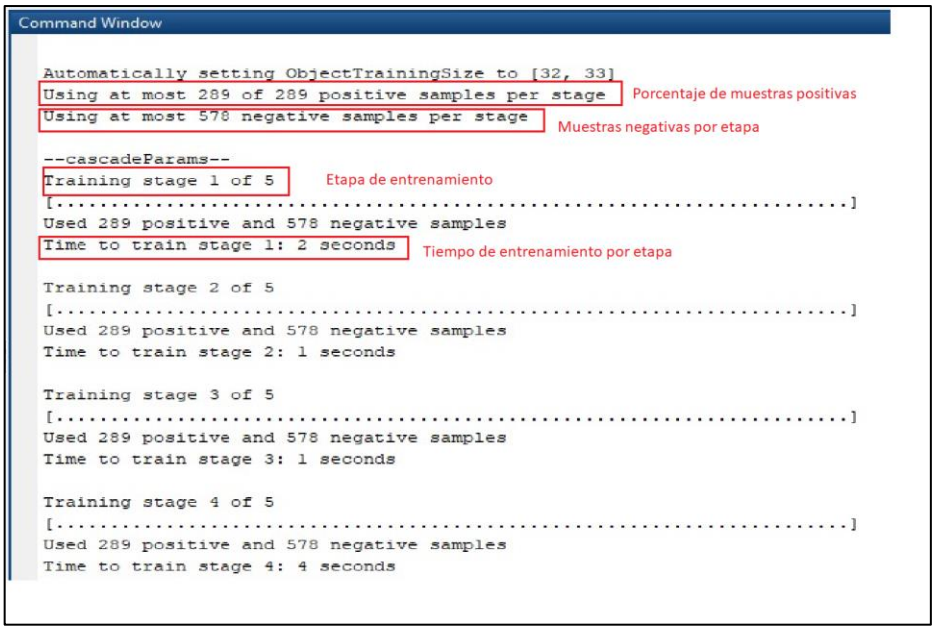


Figura 10-2: Proceso de entrenamiento de la red neuronal.

Fuente: (Diego Gordón, 2020).

Para que funcione bien, cada una de las etapas de la cascada se debe tener una tasa baja de falsos negativos. Si una etapa etiqueta incorrectamente un objeto como negativo, la clasificación se detiene y no puede corregir el error producido. Sin embargo, cada etapa puede tener una alta tasa de falsos positivos, incluso si el detector etiqueta incorrectamente un objeto como positivo, puede corregir el error en etapas posteriores del entrenamiento.

La tasa global de falsos positivos del clasificador en cascada es *fs*, donde *f* es la tasa de falsos positivos por etapa en el rango (0 1), y (*s*) es el número de etapas. De manera similar, la tasa positiva verdadera general es *ts*, donde *t* es la tasa positiva verdadera por etapa en el rango (0 1). Por lo tanto, agregar más etapas reduce la tasa positiva total falsa, pero también reduce la tasa positiva verdadera general.

2.7.2.3 Variación de parámetros de entrenamiento

Al momento de realizar nuestro entrenamiento se van a tomar las consideraciones adecuadas al establecer los parámetros:

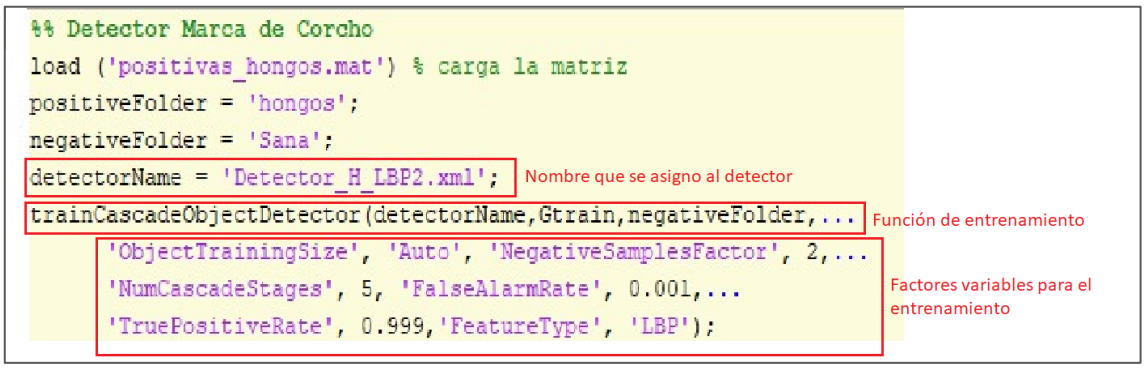


Figura 11-2: Proceso de entrenamiento de la red neuronal.

Fuente: (Diego Gordón, 2020).

En la *Figura 12-2* se muestra la sección del código que corresponde a la fase de entrenamiento del detector, la primera línea corresponde a la función que carga la matriz previamente creada por la herramienta *ImageLabeler*, la segunda línea positiveFolder corresponde al nombre de la carpeta que contiene las muestras positivas, la tercera línea contiene el nombre de la carpeta en la que van a encontrarse las imágenes negativas es decir las que no van a tener la enfermedad o alguna de las enfermedades estudiadas, la cuarta línea del código es para nombrar el detector.

La siguiente línea trainCascadeObject corresponde a la línea en donde se debe nombrar la matriz previamente cargada en este caso la nombramos como Gtrain, la siguiente sección y la más importante es la sección que podemos editar los parámetros entre estos están el factor NegativeSamplesFactor que en este caso por defecto viene con el valor 2, significa que vamos a tener una cantidad determinada de negativos y el doble de esta cantidad para positivos, la siguiente sección NumCascadeStages permite seleccionar el número de etapas de entrenamiento que vamos a aplicar, FalseAlarmRate corresponde al factor de falsos positivos este número debe acercarse en mayor cantidad al cero pero hay que tener en cuenta el número de etapas que se esté utilizando ya que si se aumentan las etapas también se debe aumentar los valores la opción TruePositiveRate y disminuir en FalseAlarmRate.

Se debe seleccionar los parámetros de la función para optimizar el número de etapas de entrenamiento, la tasa de falsos positivos, la tasa de verdaderos positivos y el tipo de

características que se utilizarán para el entrenamiento. Al momento del entrenamiento todos estos parámetros influirán de una manera positiva o negativa a los resultados obtenidos.

Tabla 5-2: Tabla de condiciones y consideraciones del entrenamiento.

Condiciones	Consideraciones
Un gran conjunto de entrenamiento	Para un buen entrenamiento el número de etapas debe ser mayor para que se establezca una tasa de falsos positivos más alta para cada etapa.
Un pequeño set de entrenamiento.	Disminuya el número de etapas y establezca una tasa de falsos positivos más baja para cada etapa.
Para reducir la probabilidad de perder un objeto.	Una alta tasa de verdaderos positivos puede evitar que logre la tasa de falsos positivos deseada por etapa, lo que aumenta la probabilidad de que el detector produzca detecciones falsas.
Para reducir el número de detecciones falsas.	Aumente el número de etapas o disminuya la tasa de falsas positivos por etapa.

Fuente: (Diego Gordón, 2020).

2.7.3 Tipos de funciones disponibles para entrenamiento

2.7.3.1 Suministro de imágenes negativas

Las muestras negativas no se especificarán explícitamente. En cambio, la función *trainCascadeObjectDetector* genera automáticamente muestras negativas de imágenes negativas proporcionadas por el usuario que no contienen objetos de interés, antes de entrenar cada nueva etapa, la función ejecuta el detector que consta de las etapas ya entrenadas en las imágenes negativas. Cualquier objeto detectado a partir de estas imágenes son falsos positivos, que se utilizan como muestras negativas. De esta manera, cada nueva etapa de la cascada está entrenada para corregir errores cometidos en etapas anteriores.

A medida que se agregan más etapas, la tasa total de falsos positivos del detector disminuye, lo que hace que la generación de muestras negativas sea más difícil. Por esta razón, es útil suministrar tantas imágenes negativas como sea posible. Para mejorar la precisión del entrenamiento, proporcione imágenes negativas que contengan fondos típicamente asociados con los objetos de interés. Además, incluya imágenes negativas que contengan objetos no similares en apariencia a los objetos de interés. Por ejemplo, si está entrenando un detector de señales de alto, incluya imágenes negativas que contengan señales de tráfico y formas similares a una señal de alto.

Figura 12-2: Imágenes negativas para el entrenamiento de la red neuronal.

Fuente: (Diego Gordón, 2020).

2.7.3.2 Suministro de muestras positivas

Para crear las muestras positivas fácilmente, se usará la aplicación *Image Labeler*, esta aplicación proporciona una manera fácil de etiquetar muestras positivas al especificar interactivamente regiones rectangulares de interés (RDI).

Además, se puede generar más muestras positivas de las existentes agregando rotación o ruido, o variando el brillo o el contraste de las imágenes ya existentes.

Figura 13-2: Imágenes positivas para el entrenamiento de la red neuronal.

Fuente: (Diego Gordón, 2020).

Existe una compensación entre las etapas de entrenamiento, entre menos etapas con una tasa de falsos positivos más baja por etapa o más etapas con una tasa de falsos positivos más alta por etapa. Las etapas con una tasa de falsos positivos más bajan son más complejas porque contienen un mayor número de resultados débiles. Las etapas con una tasa de falsos positivos más alta contienen menos resultados débiles. En general, es mejor tener una mayor cantidad de etapas simples porque en cada etapa la tasa global de falsos positivos disminuye exponencialmente.

Si la tasa de falsos positivos en cada etapa es del 50%, entonces la tasa general de falsos positivos de un clasificador en cascada con dos etapas es del 25%. Con tres etapas, se convierte en 12.5%, y así sucesivamente. Sin embargo, cuanto mayor es el número de etapas, mayor es la cantidad de datos de entrenamiento que requiere el clasificador. Además, aumentar el número de etapas aumenta la tasa de falsos negativos. Este aumento da como resultado una mayor probabilidad de rechazar una muestra positiva por error.

El entrenamiento a veces puede terminar temprano. Por ejemplo, suponga que el entrenamiento se detiene después de siete etapas, aunque establezca el parámetro número de etapas en 20. Es posible que la función no pueda generar suficientes muestras negativas. Si ejecuta la función nuevamente y establece el número de etapas en siete, no obtendrá el mismo resultado. Los resultados entre las etapas difieren porque la cantidad de muestras positivas y negativas que se utilizarán para cada etapa se recalcula para la nueva cantidad de etapas.

2.7.3.3 Tiempo de entrenamiento del detector

Entrenar un buen detector requiere cientos de muestras para el entrenamiento. El procesamiento de grandes cantidades de datos en el software puede llevar horas o incluso días. Durante el entrenamiento que se lleve a cabo, la función muestra el tiempo que llevó entrenar cada etapa en la ventana de comandos de MATLAB®. El tiempo de entrenamiento depende del tipo de función que usted haya seleccionado para el entrenamiento. Usar las funciones de Haar lleva mucho más tiempo que usar las funciones de LBP o HOG.

Se debe seleccionar la función que se adapte al tipo de detección de objetos que se necesite. *TrainCascadeObjectDetector* admite tres tipos de funciones: Haar, patrones binarios locales (LBP) e histogramas de gradientes orientados (HOG). Las funciones de Haar y LBP a menudo se usan para detectar caras porque funcionan bien para representar texturas de escala fina. Las funciones HOG a menudo se utilizan para detectar objetos. Son útiles para capturar la forma general de un objeto.

Se debe ejecutar ejecutar la función *trainCascadeObjectDetector* varias veces para ajustar los parámetros. Para ahorrar tiempo, puede usar las funciones LBP o HOG en un pequeño subconjunto de sus datos, ya que entrenar un detector con las funciones de Haar lleva mucho más tiempo de procesamiento.

2.8 Creación de un blog

Con la ayuda de esta red social se puede difundir la información de una forma clara y concisa se usó esta plataforma para realizar una pequeña síntesis de las enfermedades donde se detalla una pequeña introducción, una prevención de cada enfermedad, y una forma de controlar a la enfermedad.

Figura 14-2: Captura de página principal del blog creado.

Fuente: (Diego Gordón, 2020).

2.9 Ingreso de datos a plataforma IoT

Como primer paso para subir los datos obtenidos a la plataforma IoT fue crear un usuario en la plataforma ThingSpeak una vez creado un usuario la aplicación nos proporciona una clave de API de escritura única para cada usuario esta clave nos permite escribir datos en mi canal, con la clave API podemos enlazar Matlab con la plataforma IoT, si por alguna razón nuestra clave API fue comprometida podemos generar una nueva para asegurar nuestra integridad en los datos.

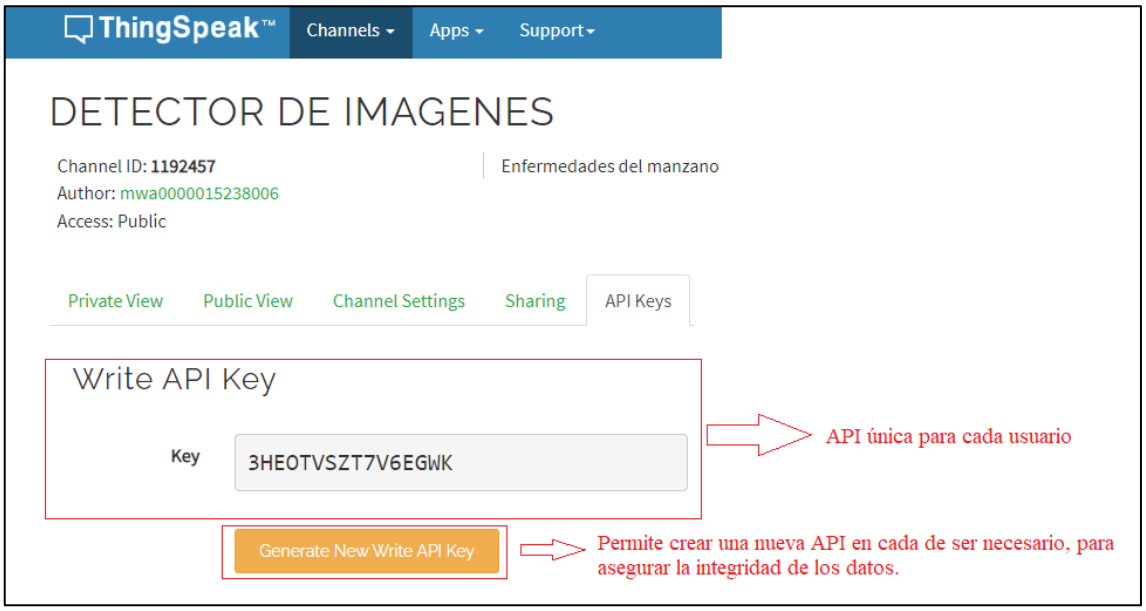


Figura 15-2: Captura de la clave API de escritura proporcionada por la plataforma IoT.

Fuente: (Diego Gordón, 2020).

De igual manera la plataforma ThingSpeak nos proporciona una clave API de lectura la cual permite que otras personas vean sus feeds y gráficos de canales privados.

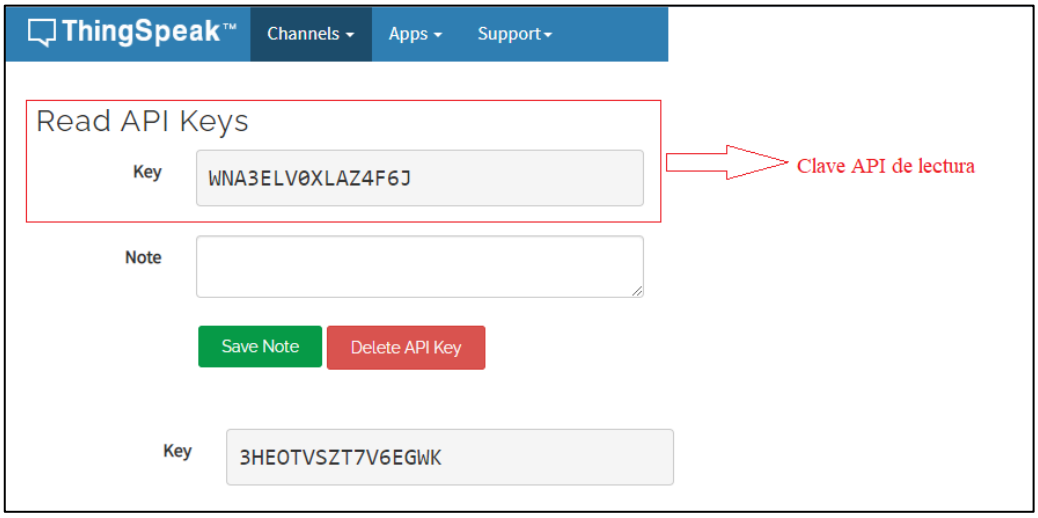


Figura 16-2: Captura de la clave API lectura proporcionada por la plataforma IoT.

Fuente: (Diego Gordón, 2020).

Para agregar las celdas en las cuales vamos a recibir los datos ThingSpeak nos proporciona la opción Channel Settings con la que nos proporcionara 8 campos en los cuales podemos agregar información según se requiera, se tomó solo 3 ya que son 3 enfermedades de manzanos las cuales vamos a realizar el análisis, el primer campo para Minadora, el segundo campo para Liquen y el tercer campo para Sarna del Manzano.

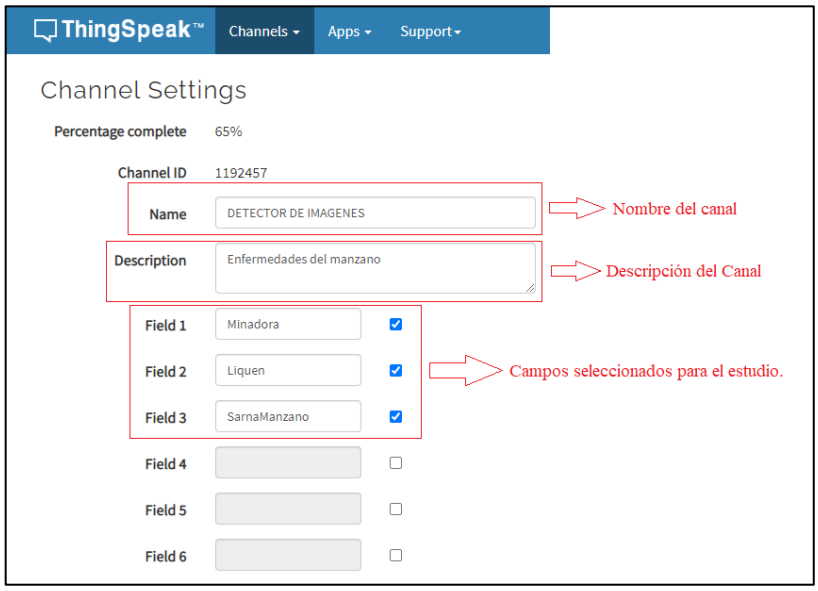


Figura 17-2: Campos creados para recibir los datos de Matlab.

Fuente: (Diego Gordón, 2020).

De igual manera la plataforma IoT de ThingSpeak nos proporciona la opción de colocar la latitud y longitud del proyecto para que las persona que visiten el canal sepan la ubicación exacta donde se desarrolló el proyecto o el área de estudio donde se llevó a cabo todo el proyecto, por lo que se colocó la ubicación exacta del huerto donde se recogieron las muestras.

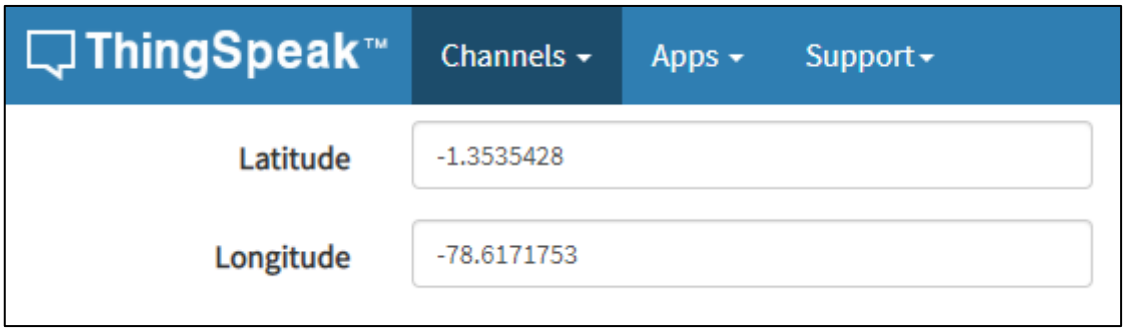


Figura 18-2: Ubicación exacta del área de estudio.

Fuente: (Diego Gordón, 2020).

Para un mejor entendimiento del canal se agregó enlace externo en el cual esta una breve reseña de la enfermedad del manzano, con los datos más importantes y un tratamiento a cada enfermedad del manzano, las personas que miren el canal de IoT adicional a la información que se obtenga de las enfermedades sepan cual es el tratamiento a seguir para que puedan dar un tratamiento rápido a las enfermedades detectadas.

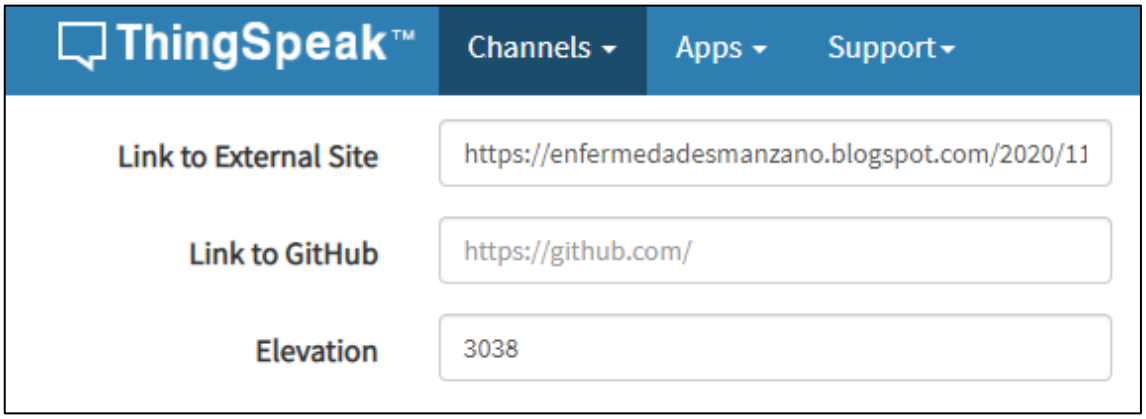


Figura 19-2: Enlace Externo a un blog.

Fuente: (Diego Gordón, 2020).

En el entorno de programación de Matlab se necesita tan solo 3 líneas de código con las cuales enlazaremos la plataforma IoT, primero necesitamos en código de identificación del canal, en la segunda línea necesitamos la llave única de escritura con la cual nos permitirá recopilar los datos en cada uno de los campos previamente creados, en la tercera línea tenemos 3 variables x, y, z las cuales corresponden los campos de la plataforma Iot.

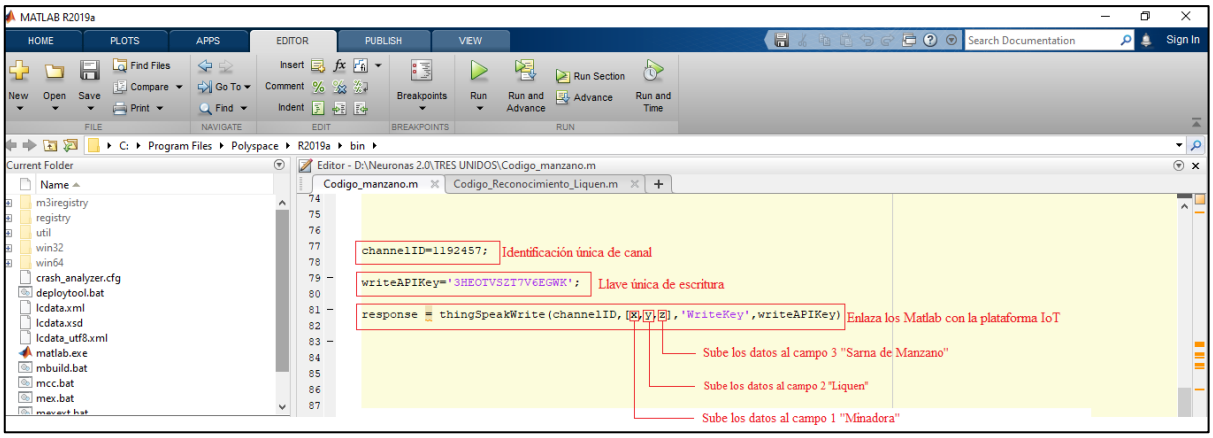


Figura 20-2: Código de Matlab para enlazar con la plataforma IoT.

Fuente: (Diego Gordón, 2020).

Una vez que se ejecute el código en Matlab los datos se irán subiendo automáticamente a la plataforma de IoT, cada vez que se obtenga una imagen positiva correspondiente a la enfermedad aparecerá con un 1 en la plataforma caso contrario como un 0, si se encuentra varias positivas simultáneamente se realizara la sumatoria de las mismas.

Figura 21-2: Datos subidos a la página IoT luego de ejecutar el Matlab.

Fuente: (Diego Gordón, 2020).

Se agrego en nuestra página de IoT 2 visualizaciones con las cuales nos permiten realizar un análisis del índice de incidencia de cada enfermedad, además se agregó una visualización donde tenemos la ubicación del área de estudio.

Figura 22-2: Vista completa de la Plataforma IoT.

Fuente: (Diego Gordón, 2020).

CAPITULO III

3 MARCO DE RESULTADOS Y DISCUSIÓN

Luego de la estructuración de la base de datos de imágenes con carpetas separadas para cada enfermedad del manzano, se obtuvo el siguiente número de muestras las cuales se detallan en la tabla 1-.3.

Tabla 1-3: Numero de muestras

Enfermedades del Manzano	Numero de muestras positivas obtenidas
Sarna del manzano	650
Minadora en círculos	826
Líquenes	1091

Fuente: (Diego Gordón, 2020).

La tabla 1-3 refleja el número de muestras obtenidas positivas de una población de 3171 muestras obtenidas mediante el VANT, todas las muestras positivas están clasificadas según la enfermedad correspondiente, obteniendo un total de 2567 muestras positivas.

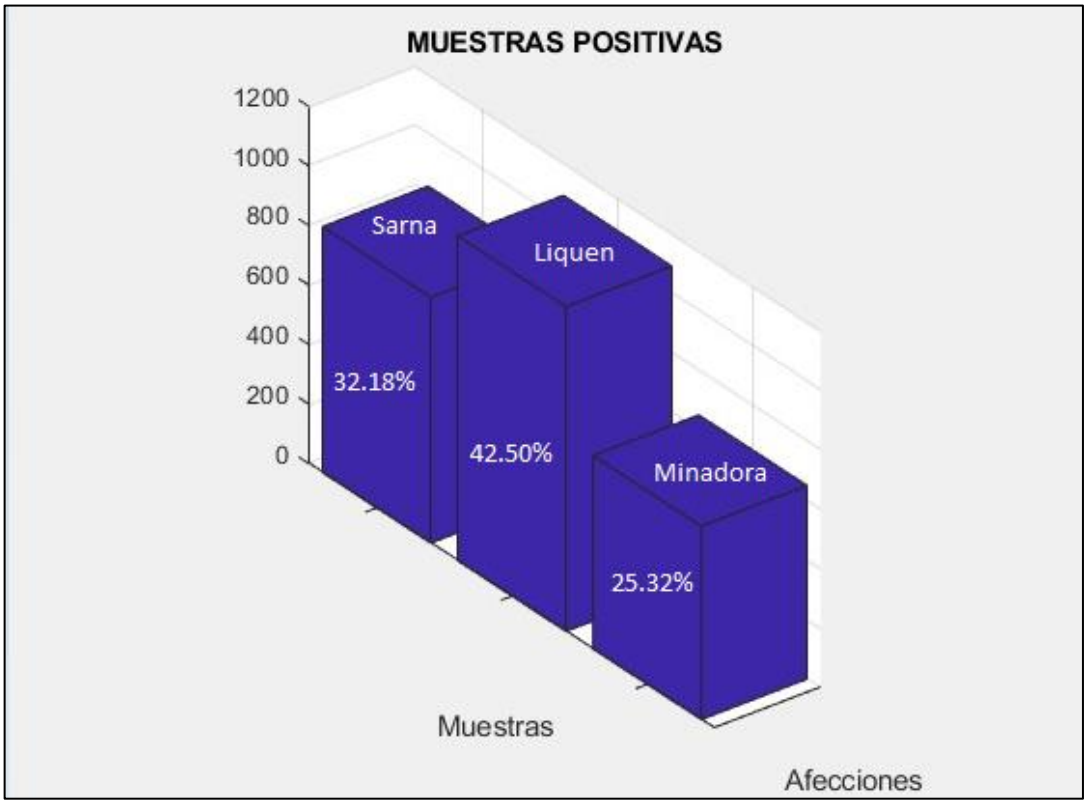


Gráfico 1-3: Resultado de las muestras obtenidas clasificadas por enfermedades.

Fuente: (Diego Gordón, 2020).

En el Grafico 1-3 tenemos el total de muestras en un esquema de barras en el cual podemos evidenciar que la enfermedad que más afecta a los huertos de manzano es la Sarna de manzano ya que se obtuvo un índice de afección de 32.18%, luego le sigue la minadora en círculos la cual tiene un índice de afección de 25.32%, y por último tenemos los líquenes los cuales tienen un índice de afección de 42.50% todos estos porcentajes corresponden al número de muestras positivas obtenidas.

3.1 Datos de presencia de enfermedades del manzano evidenciados en Matlab

En el Grafico 1-4 se muestra la detención de una afección en este caso el Liquen, la captura de Sarna de Manzano claramente se evidencia que no se realiza ninguna detección ya que en la imagen no tenemos presente ningún padecimiento correspondiente a esta enfermedad, de igual manera en Minadora no se realiza ninguna detección ya que tampoco presenta coincidencia con la enfermedad. El liquen es detectado por su coloración y textura características principales por las cuales nuestro detector identifico cada una de las muestras positivas.

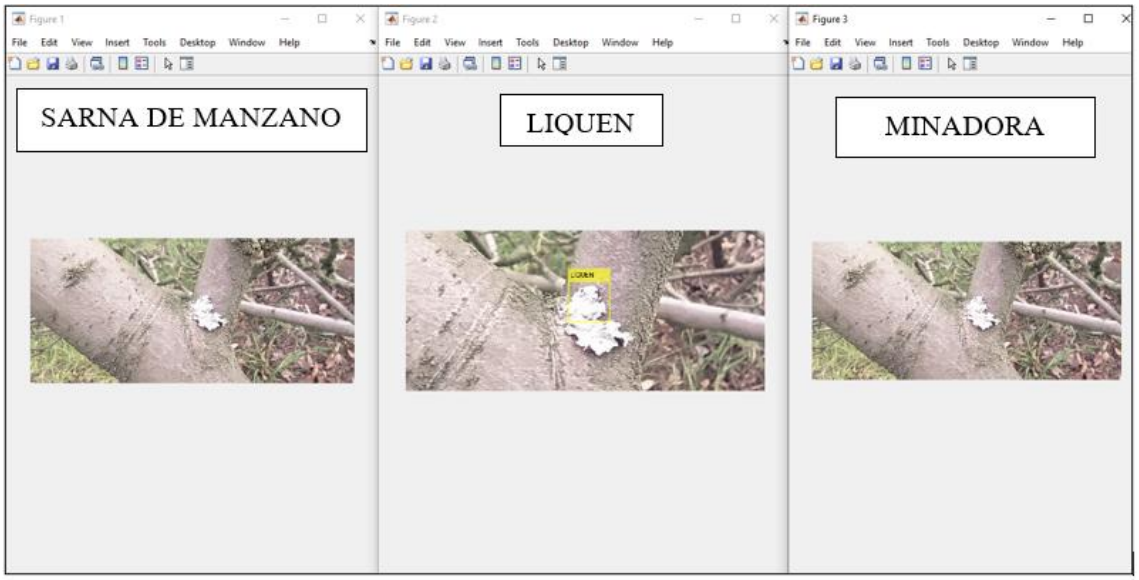


Figura 1-3: Detección de Líquenes en el manzano.

Fuente: (Diego Gordón, 2020).

En el Grafico 1-6 se muestra la detención de una afección en este caso el Minadora en Círculos, la captura de Liquen claramente se evidencia que no se realiza ninguna detección ya que en la imagen no tenemos presente ningún padecimiento correspondiente a esta enfermedad debido a que esta enfermedad solo se manifiesta en el tronco y ramificaciones principales de la planta, de igual manera en Sarna de Manzano no se realiza ninguna detección ya que tampoco presenta coincidencia con la enfermedad debido a que esta enfermedad solo se presenta en el fruto. El

patrón característico o más evidentes son las manchas foliares y costras sobre la superficie de la hoja.

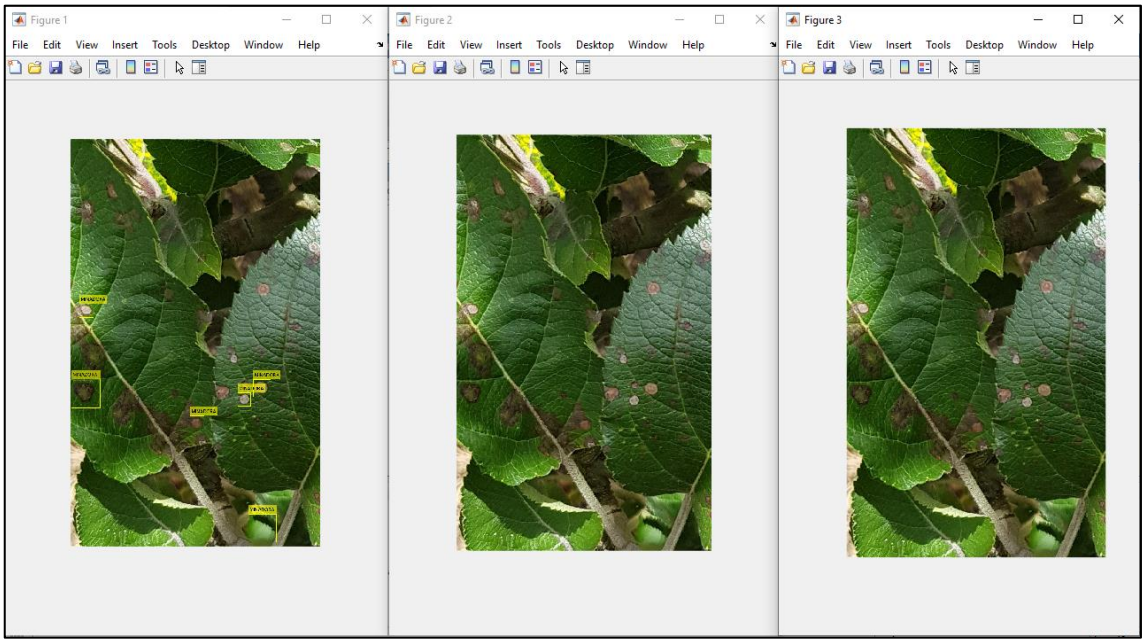


Figura 2-3: Detección de la enfermedad de manzano 'Minadora en Círculos' en la plataforma de Matlab.

Fuente: (Diego Gordón, 2020).

En el Grafico 1-8 se muestra la detención de una afección en este caso el Sarna de Manzano, la captura de Liquen claramente se evidencia que no se realiza ninguna detección ya que en la imagen no tenemos presente ningún padecimiento correspondiente a esta enfermedad, de igual manera en Sarna de Minadora en Círculos no se realiza ninguna detección ya que tampoco presenta coincidencia con la enfermedad. Esta enfermedad se caracteriza por formar unas manchas irregulares de color pardo verdoso con márgenes difusos que pueden llegar a cubrir gran parte del fruto.

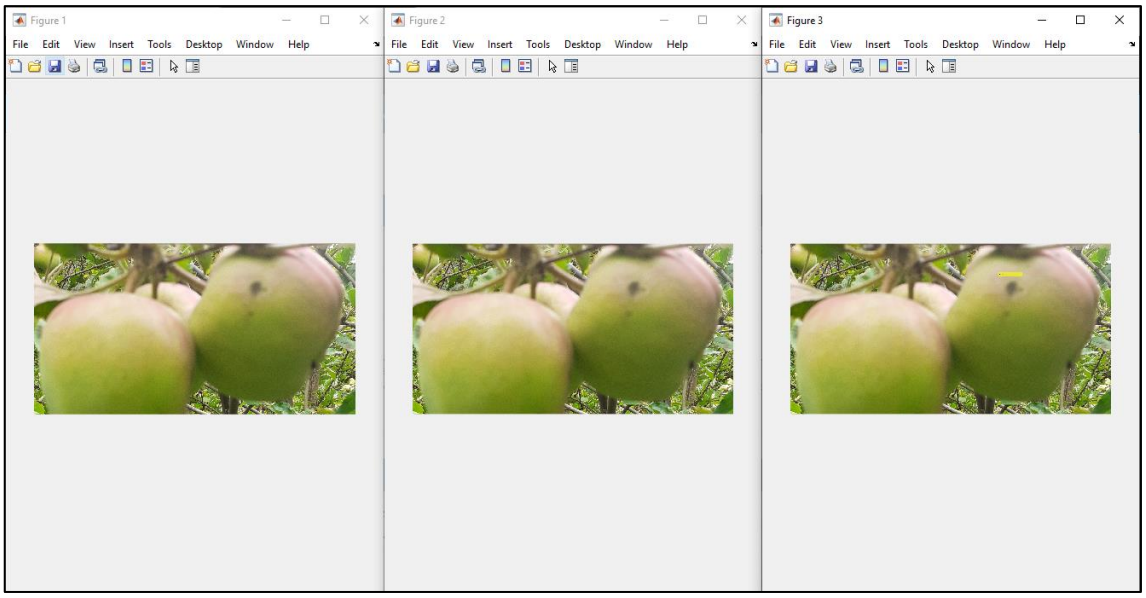


Figura 3-3: Detección de Sarna del Manzano

Fuente: (Diego Gordón, 2020).

3.2 Datos registrados en la Plataforma ThingSpeak

En el Grafico 1-5 se muestra la detención la afección Liquen en la plataforma ThingSpeak la cual nos marca el número de detecciones realizadas en la imagen cuando la muestra es positiva y con cero cuando es negativa, la captura de Sarna de Manzano, Minadora claramente se evidencia que no se realiza ninguna detección por lo que se marca con un cero, a diferencia de la detección de liquen, cabe recalcar que la actualización en la plataforma Iot, se realiza en tiempo real en conjunto con Matlab.

Figura 4-3: Detección en la plataforma de ThingSpeak. de ThingSpeak.

Fuente: (Diego Gordón, 2020).

En la Grafico 1-7 se puede visualizar claramente que se encontró 6 coincidencias con la enfermedad ‘Minadora en Círculos’, mientras que en Liquen y Sarna de Manzano muestra cero coincidencias con la enfermedad.

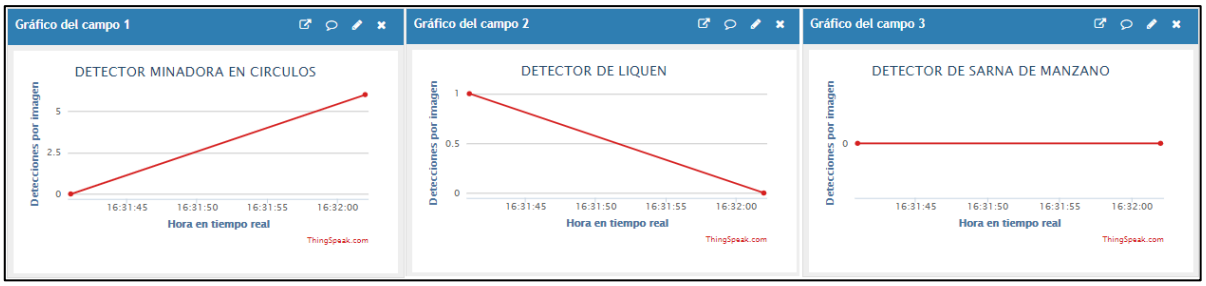


Figura 5-3: Detección de Minadora en Círculos en la plataforma de ThingSpeak

Fuente: (Diego Gordón, 2020).

En la Grafico 1-9 se puede visualizar claramente que se encontró 1 coincidencias con la enfermedad 'Sarna de Manzano', mientras que en Liquen y Minadora en Cirulos muestra cero coincidencias con la enfermedad.

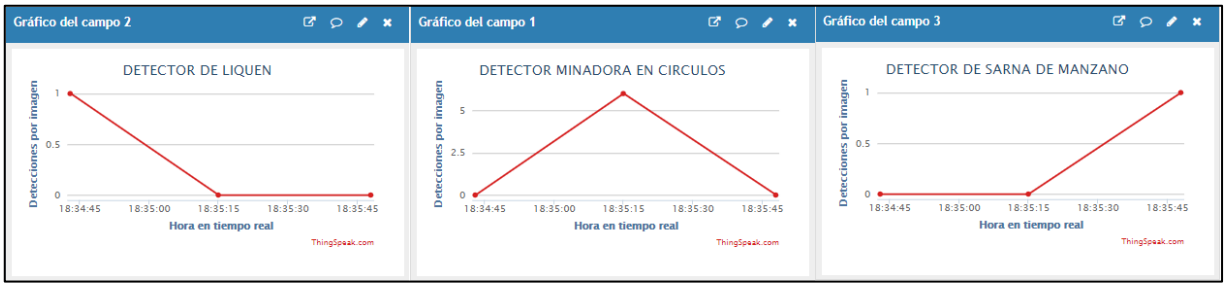


Figura 6-3: Detección de Sarna de Manzano en la plataforma de ThingSpeak

Fuente: (Diego Gordón, 2020).

3.3 Resultados

3.3.1 Resultados de la incidencia de enfermedades en toda la población estudiada

En el Grafico 1-10 podemos observar un análisis de las enfermedades del manzano en tiempo real, con el cual se determina la incidencia de cada enfermedad, claramente la que más afecta a los huertos de manzano es el liquen con el 42.50% de muestras positivas obtenidas, ya que este padecimiento se puede encontrar en cualquier parte del manzano que tenga contacto con luz solar, cabe recalcar que esta enfermedad solo se puede presentar en el tronco y en las ramificaciones principales de la planta. La segunda enfermedad de más incidencia de acuerdo a nuestro análisis y al Grafico 1-10, fue la sarna de manzano con un porcentaje 32.18% de muestras positivas obtenidas, esta enfermedad solo se presenta en el fruto. La tercera enfermedad descubierta con un índice de afección del 25.32% de muestras positivas corresponde a la Minadora en Círculos, la cual se manifiesta en las hojas del manzano. Cada una de las enfermedades se manifiesta con un patrón distintivo con el cual se logró identificar cada una de las afecciones que afectan a las diferentes partes de la planta de manzano.

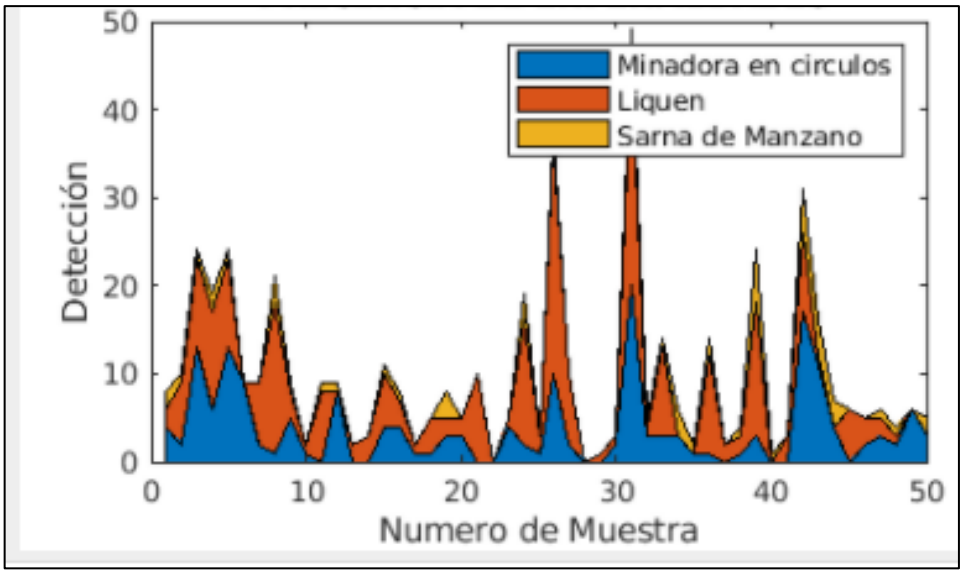


Gráfico 2-3: Análisis de la presencia de enfermedades en la población estudiada**.**

Fuente: (Diego Gordón, 2020).

En el Gráfico 1.11 se muestra el índice de incidencia de cada enfermedad respecto al número de detecciones, claramente podemos observar como la gráfica varia en el tiempo respecto al número de muestras positivas encontradas. Todos los datos de actualizan en tiempo real para realizar el análisis de las muestras.

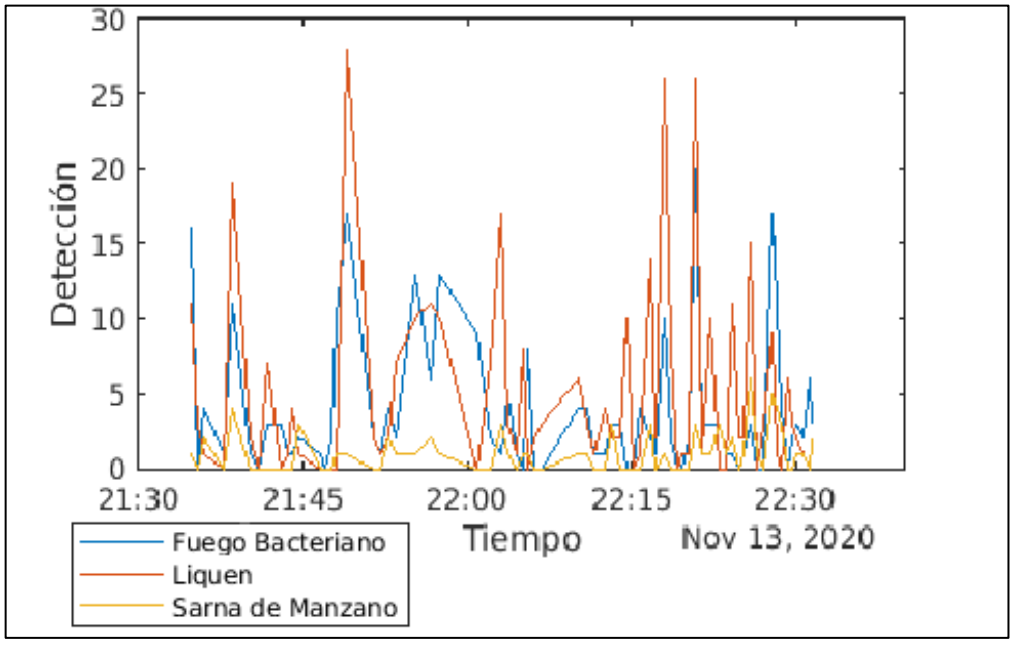


Gráfico 3-3: Análisis de la presencia de enfermedades en las muestras positivas.

Fuente: (Diego Gordón, 2020).

3.3.2 Resultados expuestos en la página web Blogger

Se creo un bloc con la herramienta web Blogger, para compartir la información más relevante del proyecto, se presenta una breve introducción sobre cada enfermedad seguido de un tratamiento preventivo, con la cual el productor podrá dar un tratamiento eficaz a cada enfermedad. El Bloc se trató de hacer lo más preciso y simple posible para un fácil entendimiento.

Figura 7-3: Bloc creado para compartir información del proyecto

Fuente: (Diego Gordón, 2020).

En nuestra página de IoT Thing Speak se agregó un botón con el cual nos direcciona a nuestro blog creado con la herramienta de Blogger donde se encontraba información detallada del proyecto, esto se lo realizo con la finalidad de simplificar el entendimiento del proyecto para que personas sin conocimiento del área de estudio comprendan de una manera más fácil.

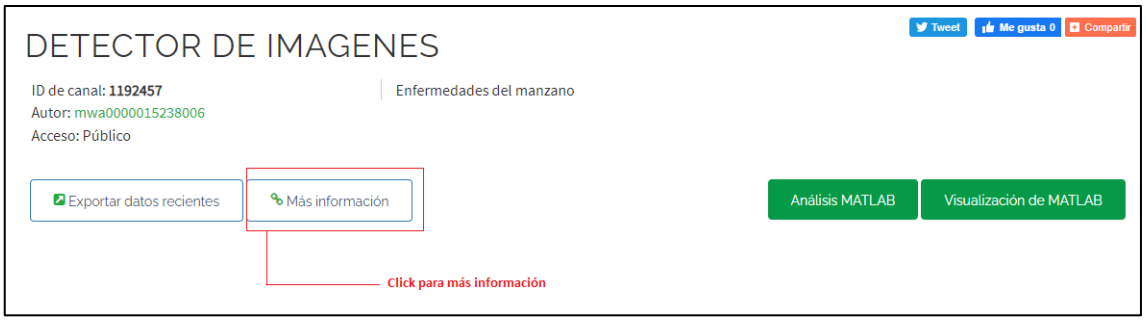


Figura 8-3: Botón con el cual se visualiza el bloc creado en la aplicación web Blogger.

Fuente: (Diego Gordón, 2020).

CONCLUSIONES

- Se logro verificar el funcionamiento de las redes neuronales artificiales ya que tuvimos resultados muy positivos en nuestro detector realizado en Matlab, se igual manera se todos los resultados obtenidos se subieron a una plataforma IoT, en los cuales se puede realizar comparaciones del índice de incidencia de las enfermedades que atacan a los árboles de manzano.

- Con la ayuda del detector realizado el Matlab se logró identificar 3 enfermedades en los árboles de manzano, una en el fruto, una en la hoja y una en el tronco de la planta las cuales son las que más afectan a los huertos en la región.

- Se realizo el diseño de un sistema de identificación de enfermedades para árboles frutales manzano, todo el diseño e implementación se lo realizo con el software Matlab, el cual incorpora herramientas que permiten el diseño de redes neuronales artificiales, la herramienta usada fue *trainCascadeObjectDetector*, con eta herramienta fue la base principal para el diseño del detector.

- Se uso una serie de algoritmos de reconocimiento de imágenes al realizar el entrenamiento de la red neuronal, todos ellos vienen incorporados en el software Matlab con el cual se realizó el reconocimiento he identificación de enfermedades del manzano.

- Se realizó todas las pruebas respectivas ya que se probó introducido imágenes positivas aleatorias, para determinar la efectividad del detector con lo cual se comprobó la efectividad de detección correspondiente a la enfermedad que aparecía en la imagen, una vez hecho esto se mandó a ejecutar todas las imágenes obtenidas, almacenadas en la base de datos.

RECOMENDACIONES

- Al momento de realizar las tomas aéreas con el VANT se deben realizar lo más cerca posible del árbol de manzano para obtener imágenes claras, con ello se tendrá una mejor respuesta en el detector.

- Al momento de realizar la selección de la parte afectada del manzano con la herramienta *trainCascadeObjectDetector* debemos ser lo más minuciosos posible ya que de eso dependerá la efectividad con la cual actuará el detector, debemos seleccionar todos los ángulos posibles de la enfermedad.

- Al momento de realizar el entrenamiento no se debe hacer uso de otras aplicaciones para el software aproveche toda la RAM del pc al momento de realizar el entrenamiento ya que esto exige unas características mínimas de 8 de RAM en el pc.

- Al momento que se exporta los datos a la página de IoT, la página nos proporciona una llave de acceso única con la cual nos asegura que los datos que subimos a la plataforma son fiables ya que el administrador es el único que puede realizar cambios en el sistema.

BIBLIOGRAFÍA

ADDATI, GASTÓN A. Y PÉREZ LANCE, GABRIEL., 2014. *Introducción a los UAV's, Drones o VANTs de uso.* Buenos Aires Estudios Macroeconómicos de Argentina : Universidad del Centro de Argentina.

AGROPECUARIAS, INSTITUTO NACIONAL DE INVESTIGACIONES, 1992. *El cultivo del Manzano en la alta del Ecuador.* Santa Catalina : INIAP.

ANDERSON, JAMES A., 2007. *REDES NEURONALES.* México : ALFAOMEGA. pág. 216.

BELLOCH, CONSUELO, 2017. *LAS TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN (T.I.C).* [En línea]. https://www.uv.es/~bellochc/pdf/pwtic1.pdf.

CALVO, JAVIER Y GUZMAN, MANUEL, 2018**.** *Machine learning, una pieza clave en la transformación de los negocios.* Barcelona : Management Solutions.

CRUZ, PEDRO PONCE, 2010. *INTELIGENCIA ARTIFICIAL CON APLICACIONES A LA INGENIERIA.* México : Alfaomega.

CUENCA, MARÍA DEL CISNE SARANGO Y BUÑAY, MANUEL ANTONIO MANIATO, 2014. *Análisis comparativo de técnicas de reconocimiento de rostros basado en modelos y en imagen usando un módulo de inteligencia artificial en Matlab.* Riobamba : ESPOCH.

GÁRATE, MARCO ANTONIO FLORES, 2014. *PRODUCCIÓN DE MANZANA.* CHUQUISACA : Imprenta IMAG.

GONZÁLEZ, PERERA Y ACOSTA, GARCÍA, 2014. *PLAGAS Y ENFERMEDADES DEL MANZANO.* Tenerife : AgroCabildo.

GUEVARA, MARTA LUCÍA y URUEÑA, WILLIAM ARDILA, 2008. *DETECCIÓN DE ROSTROS EN IMÁGENES DIGITALES USANDO CLASIFICADORES EN CASCADA.* Colombia : Universidad Tecnológica de Pereira.

HARDY, THOMAS, 2001. *Introducción a la Inteligencia Artificial.* Santiago : Universidad de Los Lagos Santiago de Chile.

HERAS, JHONSON PATRICIO CHILLO, 2019. *Evaluación de defoliantes químicos en el cultivo de manzana.* Cuenca : Universidad de Cuenca.

HOCHEL, MATEJ Y MILÁN, EMILIO GÓMEZ, 2008. *La inteligencia artificial.* Granada : Universidad de Granada.

INCIPY, 2015. *Internet of things (IoT).* s.l. : Your Digital Strategy Partner.

INESDI Y INCIPY, 2015. *Internet of things (IoT) en la transformación digital de las empresas1*. España : Your Digital Strategy Partner.

MARTÍN, PABLO RUIPÉREZ, 2014. *DISEÑO Y FABRICACIÓN DE UN DRON.* VALENCIA : UNIVERSIDAD POLITÉCNICA DE VALENCIA.

MARTÍNEZ, S Y SOLANO, E., 2010. *Blogs, Bloggers, Glogósfera* . México DF : UNIVERSIDAD IBEROAMERICANA.

MARTINSANZ, GONZALO PAJARES Y GARCÍA, JESÚS MANUEL DE LA CRUZ., 2011. *Aprendizaje Automático.* Bogotá : Ra-Ma Editorial.

MATHWORKS, 2020. Centro de ayuda e Investigación. *Centro de ayuda*. en: https://es.mathworks.com/help/matlab/learn_matlab/product-description.html.

MATICH, DAMIÁN JORGE, 2001. *Redes Neuronales y Conceptos Básicos* . Buenos Aires : Universidad Tecnológica Nacional.

MÉNDEZ, JOSÉ MORALES Y MARÍN, ROQUE, 2008. *Inteligencia Artificial, Técnicas, Métodos y Aplicaciones.* España : McGRAW.

MIGUEL CAMBRA ÁLVAREZ, ANA PALACIO, CARLOS LOZANO TOMÁS, JAIME CRESPO ALARCÓN, 2012. *EL FUEGO BACTERIANO.* ZARAGOZA : CENTRO DE SANIDAD Y CERTIFICACIÓN VEGETAL.

ORAIN, GIPUZKOA ETORKIZUNA, 2017. ENFERMEDADES DE LOS MANZANOS. *MANZANOS.* en: https://www.gipuzkoa.eus/es/web/sagarrondoak/plagas-y-enfermedades.

RAMOS, JOSÉ ANTONIO, 2016. *Generación de algoritmos.* Madrid : Universidad Carlos III de Madrid.

ROMÁN, ARMANDO, SANDOVAL, SARA Y CABELLO, MARÍA EUGENIA, 2018. *Tecnologías disruptivas de información.* Colima : Universidad de Colima.

ROSE, KAREN Y ELDRIDGE, SCOTT, 2015. *LA INTERNET DE LAS COSAS-UNA BREVE RESEÑA.* Australia : Internet Society (ISOC).

RUSSELL, STUART Y NORVIG, PETER, 1996. *Inteligencia Artificial Un Enfoque Moderno.* México : PRENTICE HALL HISPANOAMERICANA S.A.

SERRANO, JULIO y GARCÍA, ALBERTO, 2012. *INTELIGENCIA ARTIFICIAL Fundamentos, Práctica y Aplicasiones.* Madrid : Grupo RC.

SOLANO AGUILAR, GABRIELA DEL CISNE, 2018. *DISEÑO Y CONSTRUCCIÓN DE UN DISPOSITIVO PORTATIL PARA LA IDENTIFICACIÓN DE BILLETES ORIENTADO A PERSONAS CON DISCAPACIDAD VISUAL MEDIANTE EL USO DE VISIÓN ARTIFICIAL.* Riobamba : ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO.

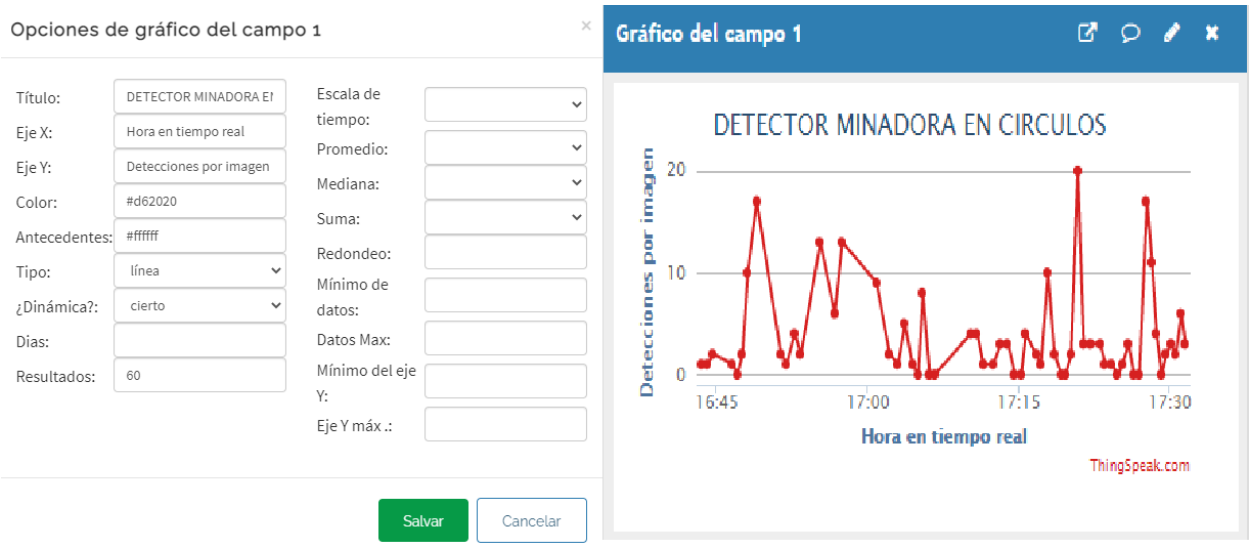
ANEXOS

ANEXO A: Código empleado en Matlab en el detector de enfermedades de manzano.

```
clc, clear all, close all
% save('positivas_corcho','Gentrenamiento_corcho') para guardar el
gentrenamiento
% trainingImageLabeler para crear las Gtrain y Gtest
%% Extraccion de imagenes de videos
% video = 'manzana.mp4';    % Nombre del video
% carpeta = 'IMG';  % Nombre de la carpeta para almacenar las imágenes
% NumeroImagenes = 20; %Número de imágenes a extraer
%
% videoobj = VideoReader(video);
% intervalo =
round(linspace(1,videoobj.NumberOfFrames,NumeroImagenes));
% for i=intervalo
%     frame = read(videoobj,i);
%     namefile = [carpeta, '\ima', num2str(i), '.jpg'];
%     imwrite(frame,namefile)
% end
%% Detector Marca de Corcho
load ('positivas_fuegobacteriano.mat') % carga la matriz
positiveFolder = 'POSITIVAS';
negativeFolder = 'NEGATIVAS';
detectorName = 'Detector_FB_LBP1.xml';
load ('positivas_líquenes.mat') % carga la matriz
positiveFolder1 = 'POSITIVAS';
negativeFolder1 = 'NEGATIVAS';
detectorName1 = 'Detector_L_LBP.xml';
load ('positivas_corcho.mat') % carga la matriz
positiveFolder2 = 'POSITIVAS';
negativeFolder2 = 'NEGATIVAS';
detectorName2 = 'Detector_MC_LBP.xml';
% trainCascadeObjectDetector(detectorName,Gtrain,negativeFolder,...
%       'ObjectTrainingSize', 'Auto', 'NegativeSamplesFactor', 2,...
%       'NumCascadeStages', 7, 'FalseAlarmRate', 0.0001,...
%       'TruePositiveRate', 0.9999,'FeatureType', 'LBP');
%% Evaluando el deteco
% Desempeño con los ejemplos de entrenamiento
% Dtrain = detectandoSobreG(detectorName, Gtruth, false);
% [F1_train, P, R] = evaluandoDetector(Gtrain,Dtrain)

% Validación con otras imagenes
% load('EmergencySigns_Test.mat');
% Dtest = detectandoSobreG(detectorName, Gtest, false);
% [F1_test, P, R] = evaluandoDetector(Gtest,Dtest)
detector = vision.CascadeObjectDetector(detectorName);
img = imread('POSITIVAS/119.jpg');
bbox1 = step(detector, img);
for k = 1:length(lee_archivos) %recorre número de archivos guardados
en el directorio
detectedImg =insertObjectAnnotation(img,'rectangle',bbox1,'MINADORA');
figure; imshow(detectedImg);
detector = vision.CascadeObjectDetector(detectorName1);
img = imread('POSITIVAS/119.jpg');
bbox2 = step(detector, img);
detectedImg = insertObjectAnnotation(img,'rectangle',bbox2,'LÍQUEN');
figure; imshow(detectedImg);
detector = vision.CascadeObjectDetector(detectorName2);
img = imread('POSITIVAS/119.jpg');
```

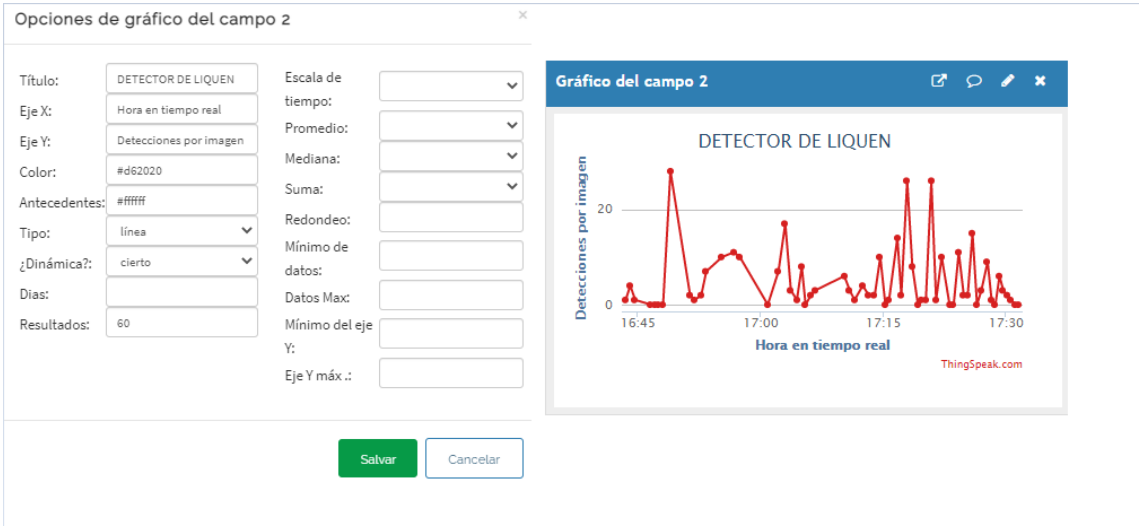
```
bbox3 = step(detector, img);
detectedImg = insertObjectAnnotation(img,'rectangle',bbox3,'SARNA');
figure; imshow(detectedImg);

%% ubicaion de la plataforma
bbox1(:,1)
F=ans
[m1,n1]=size(F)
x=m1

bbox2(:,2)
L=ans
[m2,n2]=size(L)
y=m2

bbox3(:,3)
H=ans
[m3,n3]=size(H)
z=m3
bbox(:,1)
num=ans
[m,n]=size(num)
x=m

% channelID=1192457;
% writeAPIKey='3HEOTVSZT7V6EGWK';
```

ANEXO B: Opciones de campo 1 para la toma de muestra con la página de IoT

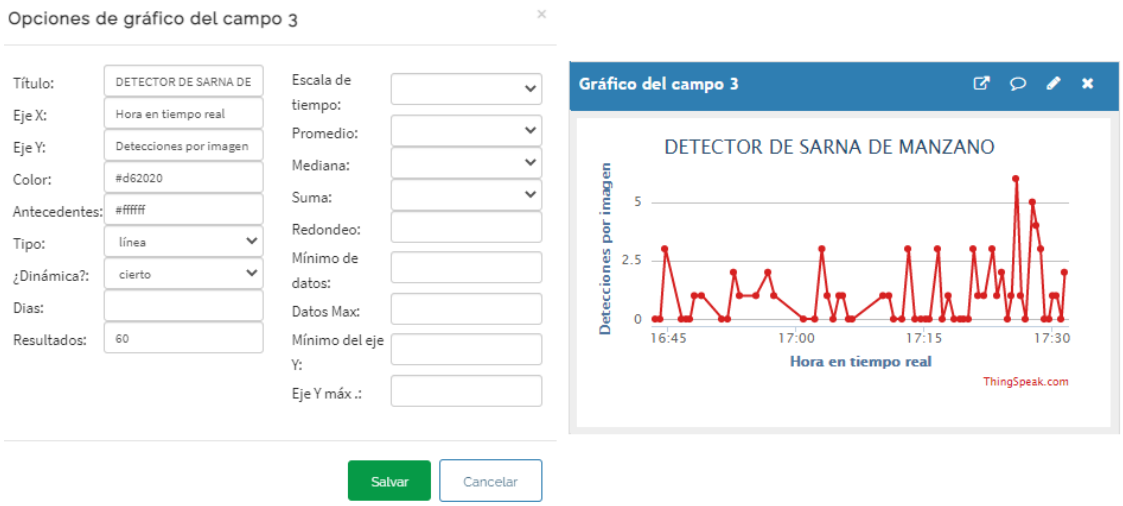


ANEXO C: Opciones de campo 2 para la toma de muestra con la página de IoT

Opciones de campo 2 para la toma de muestra con la página de IoT



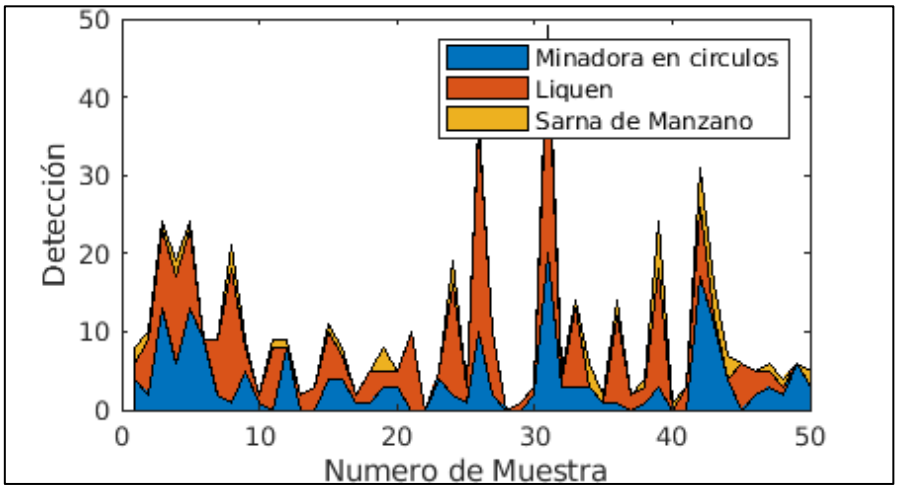
ANEXO D: Opciones de campo 3 para la toma de muestra con la página de IoT



ANEXO E: Código de la revisión por área

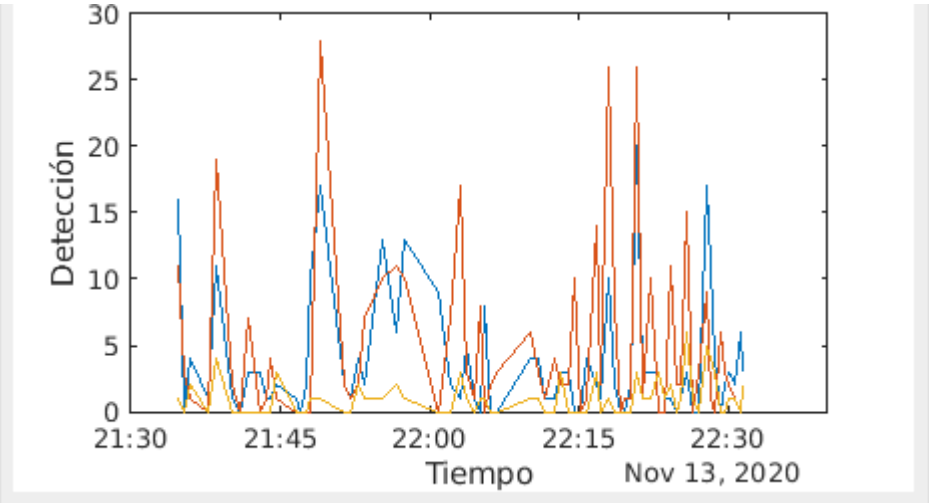
```
% Ingrese su código MATLAB a continuación
readChannelID = [1192457];


fieldID1 = [1];
fieldID2 = [2];
fieldID3 = [3];
% De clave de API de lectura de canal
% Si su canal es privado, ingrese la API de lectura
% Clave entre '' a continuación:
readAPIKey = 'WNA3ELV0XLAZ4F6J' ;


%% Leer datos


% Leer la primera variable de datos
```

```
data1 = thingSpeakRead (readChannelID, 'Field' , fieldID1, 'NumPoints' ,
50, 'ReadKey' , readAPIKey);
data2 = thingSpeakRead (readChannelID, 'Field' , fieldID2, 'NumPoints' ,
50, 'ReadKey' , readAPIKey);
data3 = thingSpeakRead (readChannelID, 'Field' , fieldID3, 'NumPoints' ,
50, 'ReadKey' , readAPIKey);
%% Procesar datos
% Concatenar las dos variables de datos
datos = [datos1, datos2, datos3];
%% Visualizar datos
área (datos);
título ( 'PRECENCIA DE ENFERMEDADES' )
xlabel ( 'Num
```




```
% Ingrese su código MATLAB a continuación

readChannelID = 1192457;


FieldID1 = [1];
FieldID2 = [2];
FieldID3 = [3];


readAPIKey = 'WNA3ELV0XLAZ4F6J' ;


[FuegoBacterianoData, tiempo] = thingSpeakRead (readChannelID, 'Fields' ,
FieldID1, 'NumPoints' , 100, 'ReadKey' , readAPIKey);


LiquenData = thingSpeakRead (readChannelID, 'Fields' , FieldID2,
'NumPoints' , 100, 'ReadKey' , readAPIKey);


SarnaManzanoData = thingSpeakRead (readChannelID, 'Fields' , FieldID3,
'NumPoints' , 100, 'ReadKey' , readAPIKey);
```

```
% plot (time, [LiquenData, FuegoBacterianoData, MarcaCorchoData],
'leyenda', ...
 % {'Liquen', 'Fuego Bacteriano', 'Marca de Corcho'});
plot (tiempo, [FuegoBacterianoData, LiquenData, SarnaManzanoData])
título ( 'COMPARACIÓN DE PRECENCIA DE ENFERMEDADES' )
xlabel ( 'Tiempo' )
ylabel ( 'Detección' )
```



ANEXO F: Ubicación del área de estudio