



POSGRADOS

Maestría en ELECTRÓNICA Y AUTOMATIZACIÓN

RPC-SO-19-No.277-2018

Opción de
titulación:

PROYECTOS DE DESARROLLO

TEMA:

DESARROLLO DE UN SISTEMA CLASIFICADOR DE
CONTAMINANTES EN UNA MUESTRA DE AGUA UTILIZANDO
INTELIGENCIA ARTIFICIAL

AUTOR:

CARLOS ALBERTO RAMIREZ QUIROGA

DIRECTOR:

VICTOR DAVID LARCO TORRES

Guayaquil - Ecuador
2022

Autor:



Carlos Alberto Ramírez Quiroga.

Ingeniero en Electricidad Especialización Potencia.
Candidato a Magíster en Electrónica y Automatización,
Mención en Informática Industrial por la Universidad
Politécnica Salesiana - Sede Guayaquil.
cramirezq@est.ups.edu.ec

Dirigido por:



Víctor David Larco Torres.

Ingeniero en Electricidad Especialización en Electrónica y
Automatización Industrial.
Magister en Administración de Empresas.
Magister en Métodos Matemáticos y Simulación Numérica
en Ingeniería.
vlarco@ups.edu.ec

Todos los derechos reservados.

“Queda prohibida, salvo excepción prevista en la Ley, cualquier forma de reproducción, distribución, comunicación pública y transformación de esta obra para fines comerciales, sin contar con autorización de los titulares de propiedad intelectual. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual. Se permite la libre difusión de este texto con fines académicos e investigativos por cualquier medio, con la debida notificación a los autores.”

DERECHOS RESERVADOS

©2022 Universidad Politécnica Salesiana.
GUAYAQUIL – ECUADOR – SUDAMÉRICA
CARLOS ALBERTO RAMÍREZ QUIROGA.

***DESARROLLO DE UN SISTEMA CLASIFICADOR DE
CONTAMINANTES EN UNA MUESTRA DE AGUA
UTILIZANDO INTELIGENCIA ARTIFICIAL***

Índice general

Índice de Figuras	V
Índice de Tablas	VII
Resumen	VIII
Abstract	VIII
1. Introducción	1
1.1. Descripción general del problema	2
1.2. Objetivos	3
1.2.1. Objetivo general	3
1.2.2. Objetivos específicos	3
1.3. Contribuciones	3
1.4. Organización del manuscrito	4
2. Marco Teórico	5
2.1. Estado del Arte	6
2.2. Definiciones Previas	12
2.2.1. Inteligencia Artificial	12
2.2.2. Visión Artificial	13
2.2.3. Aprendizaje automático o Autónomo (Machine learning)	15
2.2.4. Aprendizaje Profundo (Deep Learning)	17
2.2.5. Red neuronal Artificial (RNA).	18
2.2.6. Red neuronal convolucional	21
2.3. Formulación del Problema	36
3. Marco Metodológico	37
3.1. Diseño del experimento	38
3.2. Desarrollo	40

<i>ÍNDICE GENERAL</i>	IV
4. Resultados y Discusión	72
4.1. Evaluación Métricas utilizadas	73
4.1.1. Matriz de confusión	74
4.1.2. Tabla de Resultados de Métricas	74
4.2. Discusión de Resultados	76
5. Conclusiones y Recomendaciones	78
A. Programa completo del Clasificador	84

Índice de Figuras

2.1. La inteligencia Artificial es capaz de ver, oír y comprender . . .	12
2.2. Sistema de visión artificial	13
2.3. Diagrama de bloque de las etapas de un sistema de visión artificial	14
2.4. Tipos de Aprendizaje Automático	15
2.5. Técnicas de Aprendizaje Automático	17
2.6. Comparación de flujo de trabajo entre aprendizaje automático y profundo	18
2.7. Esquema de Red Neuronal multicapas 10-10-5	19
2.8. Arquitectura de una CNN	21
2.9. Estructura “ Capas ” CNN	22
2.10. Pasos y Desafíos para entrenar una red convolucional desde cero	23
2.11. Proceso Transfer Learning para la identificación de Objetos .	24
2.12. Comparación Arquitectura VGG19 y Residual	25
2.13. Diagrama de un bloque residual	26
2.14. Filtros separables en profundidad	27
2.15. Comparación de Estructuras tipo estándar con Mobilenet . .	28
2.16. Bloque básico Mobilenet V2	28
2.17. Ejemplo de Matriz de confusión binaria	31
2.18. Matriz de Confusion Clasificador Multiclase	32
2.19. Gráfica ROC	34
2.20. Predicciones clasificadas en orden ascendente de puntuación .	35
3.1. Raspberry Pi 4, descripción y ubicación de sockets e interfaces	39
3.2. Pantalla Pi Imager	41
3.3. Conexión correcta cable plano	41
3.4. Configuración Cámara	42
3.5. Estación de Muestreo montada	44
3.6. Ubicación Editor Thonny Python IDE	46

3.7. Código usado para facilitar la toma de muestras	47
3.8. Opciones pantalla principal de Lobe	48
3.9. Pantalla Nuevo proyecto de Lobe	49
3.10. Opciones para importar imágenes	50
3.11. Muestras agua turbia sin etiquetar ya Importadas	51
3.12. La Etapa de entrenamiento muestra sus avances en porcentaje	52
3.13. Opciones de Configuración del Proyecto	54
3.14. Resultados de predicciones	55
3.15. Opciones de visualización de resultados de la predicción	56
3.16. Ejemplo al seleccionar etiqueta individual Sólidos	57
3.17. Prueba del modelo con con Imagen fuera de la muestra Aceite .	59
3.18. Diagrama de Bloque de la Implementación.	62
3.19. Carga de librerías y métodos.	63
3.20. Creación y asignación de entradas y salidas.	64
3.21. Carga del Modelo.	64
3.22. Definición de la función take_photo.	65
3.23. Definición de la función led_select.	66
3.24. Definición de la función principal.	67
3.25. Esquema de Pines GPIO de la Raspberry	67
3.26. Esquema de conexiones del circuito, las GPIOs y Pin usados están entre paréntesis	69
3.27. Prueba Prototipo modelo 3 con Foto.	70
3.28. Prueba Prototipo modelo 3 con muestra Agua turbia.	71
4.1. Imágenes de Test con error en su predicción.	73

Índice de Tablas

4.1. Matriz de Confusión.	74
4.2. Probabilidades de muestras relevantes del Test (diferente de uno).	75
4.3. Resultado Clasificadores.	75

Resumen

El agua es uno de los recursos estratégicos insustituible e indispensable para la supervivencia y desarrollo social de la humanidad, es un elemento clave del medio ambiente global. La extracción de información dinámica sobre el agua es de gran importancia científica para la supervisión y control de su contaminación [Wang et al., 2018]. Muchos de los agentes contaminantes pueden ser observados y reconocidos a simple vista; es por esto, que se plantea el problema de identificación de objetos presentes en una muestra de agua.

Recientemente, la Inteligencia Artificial (IA) se ha desarrollado y usado en muchas aplicaciones para resolver diferentes tipos de problemas actuales, con redes neuronales y aprendizaje profundo; estas técnicas se han aplicado ampliamente para conjuntos de datos con y sin patrones, debido a que no hay un proceso de extracción de características como en otras clases de aprendizaje automático, las cuales necesitan del trabajo humano para el proceso [Kamsing et al., 2019].

Este proyecto propone desarrollar un sistema de identificación y clasificación de contaminantes en una muestra de agua, usando un modelo de aprendizaje automático pre-entrenado con una base de imágenes previamente categorizadas; obtenido mediante el uso de la herramienta de Inteligencia Artificial de Microsoft “Lobe”. En concreto con la captura de una imagen del contaminante, este será evaluado y se determinará a que grupo pertenece; para lograr el objetivo planteado el modelo será descargado en formato Tensor Flow Lite en una Raspberry Pi 4 y mediante programación en Python se hará la rutina de comparación y determinación del resultado.

Palabras Claves: Inteligencia Artificial, Aprendizaje Automático, modelo pre-entrenado, detección de objetos, reconocimiento de imagen, Lobe.

Abstract

Water is one of the irreplaceable and essential strategic resources for the survival and social development of mankind, it is a key element of the global environment. The extraction of dynamic information about water is of great scientific importance for the monitoring and control of water pollution [Wang et al., 2018]. Many of the pollutants can be seen and recognized at a glance; This is why the problem of identifying objects present in a water sample is posed.

Recently, Artificial Intelligence (AI) has been developed and used in many applications to solve different types of current problems, with neural networks and deep learning; These techniques have been widely applied to data sets with and without patterns, because there is no feature extraction process as in other types of machine learning, which need human labor for the process [Kamsing et al., 2019].

This project proposes to develop a identification and classification system of pollutants in a water sample, using a pre-trained machine learning model with a previously categorized image data set; Obtained by using Microsoft's Artificial Intelligence tool "Lobe". Specifically, by capturing a pollutant image, this will be evaluated and it will be determined to which group it belongs, to achieve this proposed objective the model will be downloaded in Tensor Flow Lite tensor format on a Raspberry Pi 4 and through programming in Python the comparison and determination routine of the result will be made.

Keywords: Artificial Intelligence, Machine Learning, pre-trained model, object detection, image recognition, Lobe

Capítulo 1

Introducción

El agua dulce es un recurso natural indispensable para la vida, este recurso imprescindible para la humanidad es limitado y no renovable; de hecho el 99,7% del agua del planeta no es utilizable a menos que pase por costosos y nocivos procedimientos que afectan el medio ambiente; es por esto que el uso adecuado y gestión del agua es un tema de mucho debate y prioritario para todos.

El problema de la polución y contaminación de afluentes de agua dulce se ha acrecentado de manera relevante durante los años recientes; por ende la calidad del agua que va hacer usada para diferentes procesos industriales, de calefacción o enfriamiento, limpieza, o por las empresas de agua potable para su tratamiento debe ser monitoreada continuamente. Como consecuencia debe haber mecanismos establecidos para probar rigurosamente la calidad del agua potable en tiempo real.

El método convencional de control de la calidad del agua. Implica su recolección de manera manual en diversas áreas, y esta muestra de agua se prueba en laboratorio. Este enfoque lleva mucho tiempo y es de costo elevado. Las metodologías actuales tienen inconvenientes como: a) Laborioso b) ausencia información de calidad de agua en tiempo real c) cobertura espacial deficiente.

Este proyecto se centra principalmente en desarrollar un sistema a nivel académico de rápida implementación y bajo costo; capaz de reconocer y clasificar los contaminantes visibles en una muestra de agua solo con la captura de su imagen, usando una herramienta de aprendizaje automático ya desarrollada (Lobe de Microsoft) e integrada con una tarjeta de desarrollo (Raspberry Pi 4); permitirá procesar las imágenes y obtener los resultados de manera casi instantánea.

1.1. Descripción general del problema

Los ríos del Ecuador no escapan a la creciente contaminación creada por el aumento de las industrias (con sus procesos y malas prácticas en tratamiento de sus desechos) y por nuevos asentamientos de familias cerca de las orillas de los afluentes.

Es así que en Junio del 2016 se produce un derrame de 45 galones de búnker [ElComercio-Redacción-Guayaquil, 2016] a 2 Km y medio de la planta "La toma" de Interagua [Ecuavisa-Redacción, 2016]; este percance ocasionó la parada de la planta por casi 48 horas dejando sin provisión de agua potable a la ciudad de Guayaquil, la detección de este incidente fue primero de tipo visual; es decir, solo se pudo detectar cuando el combustible ya estaba en las inmediaciones del canal recolector de la planta. Cabe anotar que, según una investigación realizada en ese tiempo; aproximadamente unas 100 familias de la zona se abastecen directamente del agua del río Daule [Ecuavisa-Redacción, 2016]. Si bien este fue un evento puntual; en la actualidad el sector periférico a la planta de la toma se puebla cada vez más de industrias de diferente índole, convirtiéndose en un peligro potencial en caso de no tratar correctamente el agua residual en sus procesos.

Además durante la estación invernal y/o cuando se realiza mantenimiento en la represa Daule-Peripa (aguas arriba de la toma), el nivel de turbiedad se incrementa considerablemente; teniendo que aumentar la cantidad de insumos para su clarificación y procesamiento.

Actualmente no existe una herramienta que informe esta situación con anterioridad y permita tomar los correctivos necesarios o presupuestar las cantidades de insumos a ser usados. Cabe anotar que en el proceso solo se realiza un análisis de turbiedad en la entrada de los canales de recolección en tiempo real y un análisis con muestras para laboratorio cada hora.

A fin de mejorar la efectividad y eficiencia del proceso de monitoreo del espejo de agua, previo al ingreso a los canales de recolección; así como el tiempo de respuesta ante eventos como este. Se espera plantear una solución técnicamente viable, de bajo costo y de rápida implementación a nivel de prototipo, utilizando inteligencia artificial para la clasificación de visibles contaminantes.

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar un sistema de reconocimiento e identificación utilizando modelos de aprendizaje automáticos, para determinar la presencia de diferentes agentes contaminantes en una muestra de agua.

1.2.2. Objetivos específicos

- Analizar el estado del arte sobre sistemas de identificación de contaminantes en un espejo de agua, mediante la revisión sistemática de documentos científicos con el fin de sentar las bases conceptuales y metodológicas para el desarrollo del proyecto.
- Entrenar un modelo de aprendizaje automático personalizado mediante el uso de herramientas informáticas que tome como insumo una base de imágenes para que reconozca los diferentes tipos de contaminantes presentes en una muestra de agua.
- Probar el modelo obtenido mediante el uso de muestras aleatorias para obtener su porcentaje de efectividad y mejorarlo mediante reentrenamiento.
- Crear un prototipo que con solo una imagen de la muestra permita reconocer y clasificar el tipo de contaminante presente, desplegando el modelo de aprendizaje automático en una tarjeta de desarrollo a fin de que pueda ser usado como un clasificador de agentes contaminantes.

1.3. Contribuciones

Actualmente en nuestro país el uso de las nuevas tecnologías como IoT, Big Data e Inteligencia Artificial son poco frecuentes, a pesar de su amplio campo de acción y su ya demostrada efectividad; este proyecto espera mostrar como un problema de tanta trascendencia y cotidianidad, que afecta de manera directa la vida del planeta puede ser resuelto de manera eficaz y efectiva mediante el uso de la inteligencia artificial.

Se podrá determinar los porcentajes de contaminación por tipo de agente y con estos datos prever la cantidad de materiales, insumos o equipos requeridos para dar comienzo al proceso de reducción de agentes contaminantes en los ríos, esteros y demás afluentes hídricos en las ciudades

de nuestro país, siendo la identificación de objetos en la superficie del agua el primer paso hacia la recolección y procesamiento de residuos que impactan de manera perjudicial al medio ambiente.

Emplear herramientas de Inteligencia Artificial con el fin de solucionar el problema de identificación de objetos en un espejo de agua permitirá, eliminar los inconvenientes que tienen las soluciones convencionales, como son la limitación de tiempo, personal disponible y costo elevado.

1.4. Organización del manuscrito

La estructura de este trabajo posee cinco capítulos, cada uno con temas específicos que presentan las bases teóricas y las herramientas indispensables para la comprensión de los conceptos usados en el desarrollo e implementación de este proyecto de titulación; La organización esta planteada acorde al modelo sugerido por la Universidad Politécnica Salesiana y se desglosa como sigue:

- Capítulo 1: Introducción, Descripción general del Problema, Objetivo General, Objetivos Específicos
- Capítulo 2: Marco Teórico, Estado del Arte, Definiciones previas, Formulación del problema.
- Capítulo 3: Diseño del Experimento, Desarrollo.
- Capítulo 4: Resultados y Discusión.
- Capítulo 5: Conclusiones y Recomendaciones
- Finalmente se adjunta la bibliografía correspondiente

Capítulo 2

Marco Teórico

El presente capítulo trata los conceptos básicos y fundamentos teóricos empleados en la resolución del problema propuesto. También examina el estado del arte actual de las técnicas empleadas en la supervisión de la calidad del agua e identificación de objetos.

2.1. Estado del Arte

El agua dulce es el recurso más valioso para los seres humanos. Cualquier desequilibrio en su calidad afectaría seriamente su condición de salud. [Gopavanitha and Nagaraju, 2017] Desde que la humanidad pisó el planeta el uso adecuado y la gestión del agua ha sido un tema de mucho debate y de preocupación para todos. [Praba et al., 2018]

Actualmente las empresas de servicio de agua potable enfrentan varios desafíos debido a los recursos hídricos limitados, calentamiento global, el aumento de la población y contaminación [Gopavanitha and Nagaraju, 2017]; además de la creciente capacidad industrial actual; por lo cual, la reserva mundial de agua se está volviendo perpetuamente escasa [Mhaisen et al., 2018], se estima que para la próxima década aproximadamente el 25 % de la población de la tierra vivirá en perpetua escasez de agua [Rajurkar et al., 2017]; por lo tanto se necesitan mejores metodologías en la supervisión sobre calidad del agua en tiempo real [Gopavanitha and Nagaraju, 2017].

El mundo espera cada vez más la adaptación y empleo de las técnicas recientes; a fin de que mejoren la calidad de vida y reduzcan el impacto de las actividades humanas por su conducta de consumo de recursos. [Suresh et al., 2017]; en este ámbito podemos detallar los siguientes desarrollos:

Las tecnologías de monitoreo de agua en línea han dado un progreso significativo en la supervisión de las fuentes de agua y la operación de sus plantas de tratamiento; Un modelo para un sistema con baja inversión monetaria, implementado para vigilar en tiempo real la calidad del agua, conjuntamente con el control de su flujo mediante el uso de IoT se detalla en [Gopavanitha and Nagaraju, 2017]. El sistema propuesto consta de sensores para revisar la calidad del agua y una válvula solenoide la cual que permite controlar el fluido que pasa por la tubería. Estos dispositivos son flexibles de bajo costo y alta eficiencia; están conectados al núcleo controlador de una Raspberry Pi y a un módulo IoT. Finalmente los valores supervisados son observados; y el control se lo realiza a través de Internet y también por medio de Wi-Fi para dispositivos móviles. Este sistema se puede utilizar en muchos campos como sistema de distribución, industrias y acuicultura. Este proceso de seguimiento y control se puede realizar en cualquier momento y lugar del mundo.

La mayor parte del desperdicio de agua en comunidades residenciales se produce debido al desbordamiento de los tanques de agua, fugas en tuberías y mala calidad del agua. Un enfoque de este problema es tratado en

[Praba et al., 2018] donde el objetivo del proyecto es diseñar un sistema que gestiona el nivel del agua en los tanques, así como el proporcionar una visión general y control sobre el flujo de agua para comunidades residenciales y oficinas con el uso de sensores de nivel (ultrasónico), Flujo y pH. Los datos se envían a la nube a través del módulo IoT y el usuario reconocerá el nivel del agua por medio de una aplicación de Android. Así el hardware de integración se compone de una placa Arduino con un dispositivo IC que recopila las lecturas y transfiere al módulo Iot.

En [Mhaisen et al., 2018] se intenta brindar a los usuarios una conciencia completa sobre sus consumos de agua, lo cual ayudará enormemente a minimizar el desperdicio de esta, el trabajo presenta el diseño de un sistema novedoso y autónomo; que tiene como objetivo presentar una monitorización inteligente y autoalimentada. Este sistema aprovecha el IoT y la computación en la nube; consiste de un dispositivo IoT que se puede instalar en cualquier fuente de agua, una aplicación de nube que permite tomar datos de los sensores; además de una aplicación móvil para visualizar el consumo de agua en cada fuente monitoreada. El sistema le da al usuario la capacidad de identificar la ubicación y la hora del uso excesivo y las fugas en un teléfono móvil. Los resultados experimentales confirman la premisa autoalimentada de la solución. Se desarrolla un prototipo de aplicación móvil y el backend que es probado.

Con el rápido desarrollo de la tecnología de teledetección, las imágenes de alta resolución han sido una fuente de datos importante para algunas aplicaciones como la gestión de desastres, el barrido urbano y la extracción de huellas de edificios. La extracción de imágenes por medio de la teledetección a fin de obtener la red de carreteras es presentado en [Xu et al., 2018]; su principal contribución es explorar una técnica alternativa, que combina el filtro y aprendizaje profundo; preprocesando la imagen por el filtro gaussiano, esto es útil para eliminar el ruido de clase sal y pimienta mejorando así el rendimiento de la extracción de carreteras. La red neuronal profunda (ResNet) funciona bien en el etiquetado de carreteras de diferentes escalas para los múltiples modelos, que optimizan los resultados de la clasificación. El área del estudio corresponde a Las Vegas-EEUU, la mayoría de las carreteras se han extraído con éxito; los resultados mostraron la efectividad y viabilidad del marco propuesto. Se utiliza la Precisión general, sensibilidad y Puntuación F1 como métricas de evaluación del modelo.

La teledetección junto a la visión por computadora (identificación) desarrollada con redes neuronales profundas se utilizan para el mapeo de ciudades [Yang et al., 2018]; en este trabajo se revisa los métodos de CNN de etiquetado de píxeles y su consideración para la extracción de edificios a gran escala. Se demuestra que las CNN profundas pueden beneficiarse de la noción de modelos de clasificación previamente entrenados y proceder a generar resultado de segmentación a nivel de píxeles. Además, propone un enfoque que está diseñado para delinear mejor los contornos de edificios y explorar una banda espectral adicional, que no se puede incorporar directamente en modelos preentrenados sin más adaptación y pesos de aprendizaje que los de la capa de entrada. Se utiliza métricas de validación cruzada para llevar a cabo un análisis experimental extenso con las CNN del estado del arte actual para la tarea de etiquetado de píxeles para la extracción de edificios.

La detección de paneles solares fotovoltaicos utilizando una red neuronal convolucional pre-entrenada con imágenes aéreas de alta definición es tratada en [Malof et al., 2017], En este trabajo se considera el problema de desarrollar algoritmos que identifiquen automáticamente los paneles, ya que estos ofrecen potencialmente una solución más rápida y económica para recopilar información fotovoltaica (PV) a pequeña escala, como su ubicación, capacidad y la energía que producen. Se Propone una arquitectura de CNN que está inspirada en los diseños de VGG (Visual Geometry Group); que consta de dos tipos de módulos: los VGG(x) y los de neurona completamente conectada FC(y); luego se mide su rendimiento en la detección en el mismo conjunto de datos. Los resultados indican que la CNN produce mejoras sustanciales de rendimiento sobre resultados anteriores. También se investiga los enfoques recientemente populares de pre-entrenamiento para CNN entre otras aplicaciones.

En los últimos años, con la mejora continua en la ciencia de la teledetección y la aparición de algunos nuevos algoritmos de aprendizaje automático, la investigación sobre el monitoreo con teledetección de la calidad del agua se desarrolla gradualmente de cualitativa a cuantitativa [Ding et al., 2016]. Esta tiene ventajas tales como amplio rango de supervisión, alta velocidad, bajo costo y conveniente para el monitoreo periódico continuo. Es así que pueden descubrir fuentes de contaminación y características de contaminantes que son difíciles de revelar por algunos métodos convencionales. En esta investigación con la ayuda de la tecnología de detección remota y la plataforma GIS (Sistema de información Geográfica), la sección Shenyang

del río Hun se toma como área de investigación, el modelo de inversión que es adecuado para la calidad del agua del río Hun es construido para parámetros de calidad del agua como nitrógeno amoniacal, demanda química de oxígeno (DQO) dentro Hun River, la valoración y análisis de la pureza del agua sobre la tendencia a la variación del cuerpo de agua en la sección Shenyang del río Hun se realiza mediante el resultado de inversión.

En [Wang et al., 2018] se establece un modelo de inversión de los indicadores que determinan la calidad del agua en capturas de teledetección, en función del cual se analiza el estado de la contaminación del agua. Primero, la información espectral del agua en el área estudiada se extrae de imágenes de teledetección. Luego, se modela la relación entre la reflectancia espectral y los parámetros de calidad del agua.

Se discute modelos para la estimación de parámetros de fuga de petróleo en [Gautama et al., 2016, Li et al., 2017], este trabajo desarrolla un marco novedoso para la asimilación parámetros de fuga de aceite, incluidos los datos de inicio de fuga y la duración; junto con los factores del viento y la corriente a la deriva, desde un SAR (radar de apertura sintética) abordo de un satélite. Se centra en la asimilación de las observaciones del SAR en un modelo de trayectoria lagrangiana bidimensional. Se comparan las trayectorias de petróleo en superficie simuladas a las observaciones satelitales, en los ciclos de pronóstico subsiguientes para las pruebas de velocidad mediante la estimación de los mejores parámetros de fuga de derrames de petróleo.

Un método de reconocimiento en un derrame de petróleo desde imágenes SAR basado en una red neuronal convolucional es presentado por [Wang et al., 2018]; en donde las características de la categoría son extraídas automáticamente y evita la no estandarización de los métodos de extracción manual. La imagen original de SAR del derrame de petróleo se filtra y eliminado el ruido, antes de que se introduzca en la red CNN, y la extracción de características se realiza utilizando el modelo de la CNN. Finalmente, las características están clasificadas por capas Soft-max. Los resultados de la identificación demuestran que el El método propuesto tiene una alta precisión en la identificación de "manchas de petróleo." imágenes de "manchas similares al petróleo ".

La teledetección usada en la monitorización sobre la calidad del agua tiene entre sus ventajas lo siguiente: Se puede usar en áreas relativamente amplias, es de gran velocidad y bajo costo. Pero solo conveniente para el monitoreo periódico[Ding et al., 2016], ya que debido a la influencia de las nubes, el clima y la limitación del período de revisión de satélites, solo se pueden obtener datos de muestra limitados[Yang et al., 2019]. Este problema no se presenta usando la identificación de objetos.

El uso de tecnologías para identificación de objetos usando imágenes y redes neuronales pre-entrenadas, tiene un amplio uso. Como la detección de defectos en tarjetas electrónicas (PCB), pudiendo determinar incluso los tipos de fallas que se podrían ocasionar de acuerdo al área donde se encuentran las mismas; como se menciona en [Deng et al., 2018], donde se diseña un sistema de categorización automática que puede reconocer e identificar diferentes defectos en tiempo real, basándose en algoritmos de aprendizaje profundo. El uso de la CNN para clasificar defectos y la alta tasa de reconocimiento de defectos, no solo disminuiría la tasa de falsas alarmas de la máquina AOI (automatic optical Inspection), también disminuye la carga de trabajo del personal.

La clasificación de imágenes EGG (electroencefalograma) se trata en [Carvalho et al., 2017]. En este documento se propone un nuevo enfoque para la clasificación de objetivos a alcanzar antes del inicio del movimiento, durante alcance guiado visualmente en el espacio 3D, combina el poder discriminante de la electroencefalografía bidimensional (EEG) (es decir, imágenes EEG) construidas a partir de épocas cortas, con el capacidades de extracción y clasificación de características del aprendizaje profundo (DL), como las redes neuronales convolucionales (CNN). Los movimientos de alcance se realizan en cuatro direcciones: izquierda, derecha, arriba y abajo. Para permitir alcanzar movimientos más naturales, se explora el uso de la realidad Virtual (VR) a fin de construir una configuración experimental; que le permita al sujeto realizarlos a su propio ritmo en el espacio 3D mientras está de pie.

La identificación de objetos es un problema vigente en la actualidad, por ejemplo la identificación automática de un avión en imágenes satelitales puede darnos patrones de actividad de tráfico para monitorear aeropuertos, e incluyendo problemas de inteligencia de defensa. En [Kamsing et al., 2019] se propone la implementación de redes neuronales convolucionales (CNN) con el fin clasificar un avión en un conjunto de datos, se establece

un modelo pre-entrenado y de aprendizaje de transferencia; a fin de superar la limitación de los recursos de computación, se agrega una nueva capa superior completamente conectada, para identificar las nuevas clases de aviones y volver entrenarla.

Los avances en tecnologías de semiconductores en años recientes han impulsado el desarrollo de sensores de imágenes de bajo costo en el mercado y también han hecho que tecnologías como UAV (unmanned aerial vehicle) y drones, sean más asequibles. Estas plataformas aéreas están equipados con potentes sensores de imágenes, que pueden ser utilizado para la vigilancia aérea y adquisición de imágenes. Las imágenes adquiridas de estos drones se pueden usar para detectar vehículos y formar una inferencia cuantitativa sobre el tráfico, en cualquier lugar dado. El diseño de un sistema totalmente automatizado para la detección de vehículos a partir de imágenes aéreas se trata en [Mohan et al., 2018], dado que el conjunto de datos es pequeño y requiere que la red neuronal sea entrenada desde cero, se prefiere una red fácilmente entrenable (se compara entre Alexnet y VGG 16).

Un método de detección de objetos flotantes en superficie en tiempo real basado en aprendizaje profundo se trata en [Shanhua et al., 2020]; para resolver el problema de que el objeto flotante en la superficie necesita inspección manual la cual requiere mucho tiempo y trabajo, además que la información de los únicos medios de monitoreo no es completa, este trabajo propone un sistema conjunto integrado de monitoreo y detección, que puede supervisar la información de la imagen de vídeo en varios escenarios. Que alerta y elimina automáticamente. Basado en el algoritmo de detección de objetos flotantes de superficie mediante videovigilancia, el marco darknet se utiliza para establecer una red de aprendizaje profundo, y el algoritmo de detección mejorado YOLOv3 está diseñado para resolver el problema de que la basura flota en la superficie del agua que fluye rápidamente

Por todo lo expresado; se estima como una solución viable, realizar el análisis de reconocimiento y clasificación de contaminantes del agua haciendo uso de las imágenes previamente almacenadas en una base de datos a través de inteligencia artificial ya desarrollada; las cuales servirán de insumo para el entrenamiento de un modelo de aprendizaje automático, que mediante un muestreo aleatorio (utilizado para su comprobación) verifique el grado de certeza y eficiencia en clasificación de los contaminantes presente en la muestra de agua.

2.2. Definiciones Previas

2.2.1. Inteligencia Artificial

Es una rama de las ciencias computacionales que se ocupa de los símbolos y métodos no algorítmicos para la resolución de problemas. [Ponce and Herrera, 2010]

Podemos decir que es la parte de la informática preocupada por el diseño de sistemas computacionales; que exhiben características que asociamos con la inteligencia en el comportamiento humano (comprensión del lenguaje, aprender razonando, resolver problemas, etc.) [Kordon, 2010]. Esta idea se plasma en la figura 2.1.

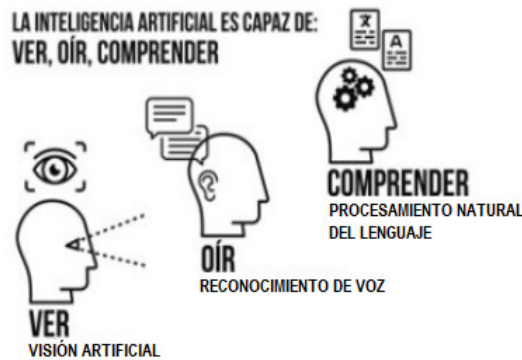


Figura 2.1: La inteligencia Artificial es capaz de ver, oír y comprender

Fuente: [Rouhiaien, 2018]

Se la vincula con la evaluación y elaboración de sistemas artificiales autónomos; los cuales son susceptibles de mostrar un funcionamiento inteligente. Para determinar que un agente actúa inteligentemente, este tiene que notar las características de su entorno, escoger y planear sus objetivos, obrar en procura de la consecución de estos (usando principios de racionalidad); además de interactuar con otros actores inteligentes, sean estos artificiales o humanos. [Palma and Marín, 2008]

Concluimos entonces que es la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones; tal y como lo haría un ser humano. [Rouhiaien, 2018]

2.2.2. Visión Artificial

La visión artificial o por computador es la ciencia y la tecnología que permite a las “máquinas” ver; extraer información de las imágenes digitales, resolver alguna tarea o entender la escena que están visionando. [i [García, 2012](#)]

Es un campo que incluye métodos para adquirir, procesar, analizar y comprender imágenes del mundo real y así producir información numérica o simbólica, por ejemplo, en forma de decisiones [[Deirdre, 2016](#)]; en la figura 2.2 se muestran los componentes de un sistema de visión artificial.

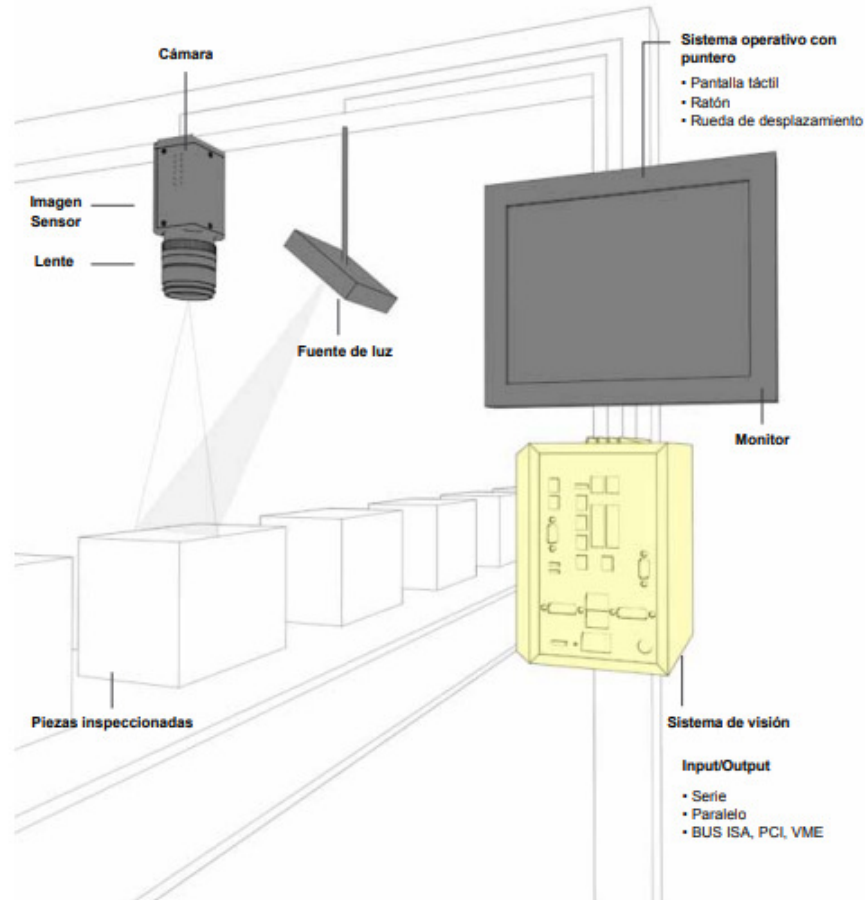


Figura 2.2: Sistema de visión artificial

Fuente: [[Cognex Corporation, 2016](#)]

Este campo interdisciplinario simula y automatiza los elementos de la visión humana utilizando sensores, computadoras y algoritmos de aprendizaje autónomo, que es la teoría que subyace en la habilidad de los sistemas de inteligencia artificial para ver y comprender su entorno [DeepAI, 2019a]. Un sistema de visión artificial está compuesto por las siguientes etapas:

- Adquisición de Imágenes.
- Pre procesamiento.
- Segmentación.
- Representación y descripción (parametrización).
- Reconocimiento e interpretación (Clasificación).

En la figura 2.3 se muestra estas etapas en un esquema de bloque.

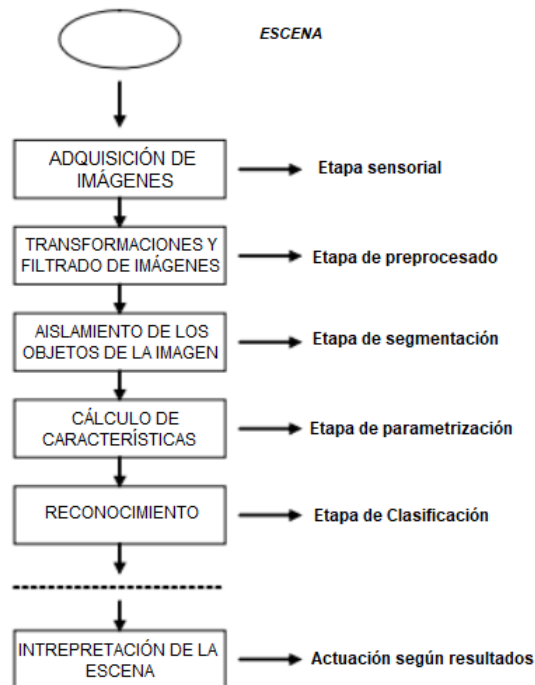


Figura 2.3: Diagrama de bloque de las etapas de un sistema de visión artificial

Fuente: [Ordieres and Martínez, 2006]

2.2.3. Aprendizaje automático o Autónomo (Machine learning)

Es un campo de la inteligencia artificial que se centra en enseñar a los ordenadores a realizar lo que es innato de las personas y animales: adquirir conocimiento por medio de la experiencia.

Los algoritmos de aprendizaje automático emplean procedimientos informáticos con el fin de “aprender” información directamente de los datos sin valerse de una fórmula preestablecida como patrón. Estos robustecen de forma adaptativa su desempeño cuando la cantidad de ejemplos disponibles para el aprendizaje se incrementa. [Mathworks, 2020]

El aprendizaje automático utiliza tres tipos de técnicas, las cuales se describen en la figura 2.4 y son: **Aprendizaje supervisado**, el cual entrena un modelo con datos de entrada y salida conocidos para que pueda predecir los resultados futuros. **Aprendizaje no supervisado**, que encuentra patrones ocultos o estructuras intrínsecas en los datos de entrada [Mathworks, 2020]; finalmente, en el **aprendizaje por refuerzo**, los algoritmos aprenden del experimento. Es decir, debemos proporcionar «un refuerzo positivo» siempre que la respuesta sea correcta. Esta manera en que los algoritmos aprenden se asemeja con la de adiestramiento de animales, por ejemplo cuando le damos «recompensas» a un perro por aprender a sentarse. [Rouhiaien, 2018]

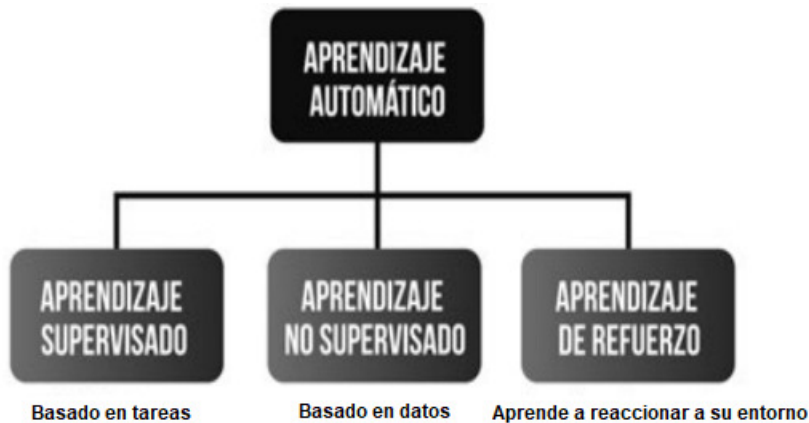


Figura 2.4: Tipos de Aprendizaje Automático

Fuente: [Rouhiaien, 2018]

El propósito del aprendizaje automático supervisado consiste en construir una estructura (modelo); que realice pronósticos fundamentados en los datos con la existencia de incertidumbre. El algoritmo toma un conjunto de datos de entrada y sus respuestas conocidas (salidas), con estas entrena un modelo para generar predicciones razonables de la respuesta a nuevos datos (muestras). Usa técnicas de clasificación y regresión para desarrollar modelos predictivos.

- Las técnicas de clasificación predicen respuestas discretas, por ejemplo, si un correo electrónico es genuino o spam, o si un tumor es canceroso o benigno. Los modelos de clasificación ordenan los datos de entrada en categorías; las aplicaciones típicas incluyen imágenes médicas, reconocimiento de voz y calificación crediticia.
- Las técnicas de regresión pronostican respuestas continuas, como: variaciones en la temperatura o alteraciones en el consumo de energía. Su uso tradicional abarca aplicaciones de distribución eléctrica, estimación de carga y algoritmos de tráfico comercial.

El aprendizaje no supervisado identifica diseños encubiertos o esquemas inherentes de los datos. Este es usado con el fin de obtener deducciones desde el conjuntos de datos, que constan de elementos de entrada y que no tienen sus respuestas categorizadas. La agrupación es la aplicación de aprendizaje no supervisado de mayor uso, se la emplea en el estudio experimental de los datos a fin de hallar sus “patrones ocultos” o la asociación presente en estos. Entre las utilidades más frecuentes de la agrupación están la evaluación de cadena de genes, la exploración del mercado y el reconocimiento objetos [Mathworks, 2020]; en la figura 2.5 podemos apreciar el enfoque para el uso apropiado de los métodos de aprendizaje supervisado y no supervisado.

El aprendizaje por refuerzo supone que la retroinformación dada al sistema se ofrece a manera de premios y penalizaciones, en vez de declararla expresamente como un “acierto” o “desacierto”. Esto aplica si es sustancial dar con la respuesta correcta, así también si es prioritario identificarla de forma ágil.

En consecuencia, un factor fundamental del aprendizaje por refuerzo es dar con el balance entre “exploración” y “explotación”. Evaluar ¿Cuánto más debería “explorar” el programa para encontrar información adicional, en vez de beneficiarse de la ya obtenida? Al “recompensar” al elemento del aprendizaje por actuar de una forma conveniente, el programa puede maximizar la relación entre la exploración y explotación y así obtener un mayor balance entre ellas.[DeepAI, 2019b]

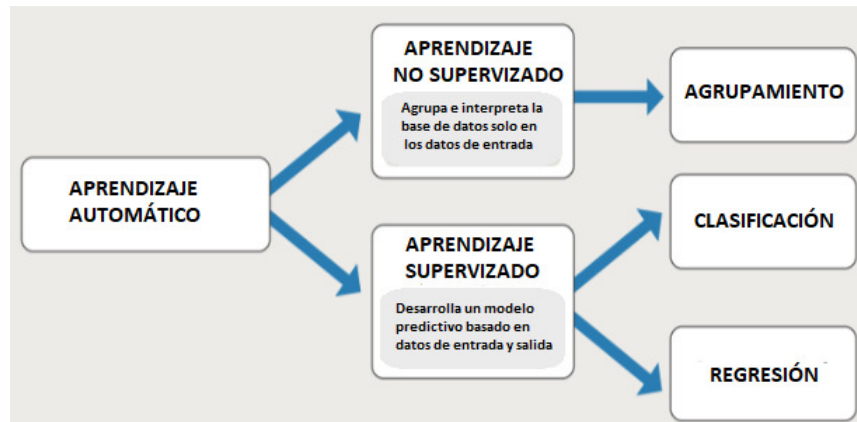


Figura 2.5: Técnicas de Aprendizaje Automático

Fuente: [Mathworks, 2020]

2.2.4. Aprendizaje Profundo (Deep Learning)

Hace referencia a una gama de algoritmos de aprendizaje automático; los cuales, utilizan de manera exhaustiva redes neuronales artificiales. En una charla técnica de Google del 2016, Jeff Dean define en los algoritmos de aprendizaje profundo el empleo de redes neuronales altamente densas, aquí el término "profundo" tiene que ver con la cantidad de fases (capas) u operaciones, presentes desde la entrada hasta la salida de la red. En tanto que la capacidad computacional reduce su costos, los algoritmos de aprendizaje para las diferentes aplicaciones actuales se vuelven "más profundos". [DeepAI, 2019b]

Es una herramienta de aprendizaje automático que utiliza múltiples capas de procesamiento no lineal para aprender representaciones de características directamente de los datos, se utiliza para resolver problemas en los campos de visión artificial, procesamiento del lenguaje natural o robótica. En otras palabras es un conjunto de algoritmos de aprendizaje automático que llevan a cabo un aprendizaje de principio a fin; es decir, le damos datos en bruto y una tarea a realizar a el algoritmo, por ejemplo: clasificación de imágenes, texto o sonido y el algoritmo aprende por si solo a clasificarlo correctamente de modo automático. [García, 2016]

Flujo de trabajo clásico (Machine learning:) Comienza trabajando con un conjunto de imágenes y en primer lugar llevaremos a cabo una extracción manual de las características de estas, aquí podemos usar una

gran variedad de métodos y técnicas. Una vez extraídas las características podemos proceder a su clasificación, recurriendo a métodos tales como: máquinas de vector soporte, k-vecinos más próximos, árboles de decisión, etc.

Flujo de trabajo por medio de deep learning: Primero las imágenes se pasan por una red neuronal convolucional (CNN), que por si sola aprenderá las características directamente de las imágenes y clasificará las mismas; de este modo la red neuronal convolucional, hará lo que se conoce como un aprendizaje de principio a fin, llevando a cabo el aprendizaje de características como la clasificación. En la figura 2.6 se muestra un esquema comparativo entre el flujo de trabajo el tradicional y el de aprendizaje profundo.

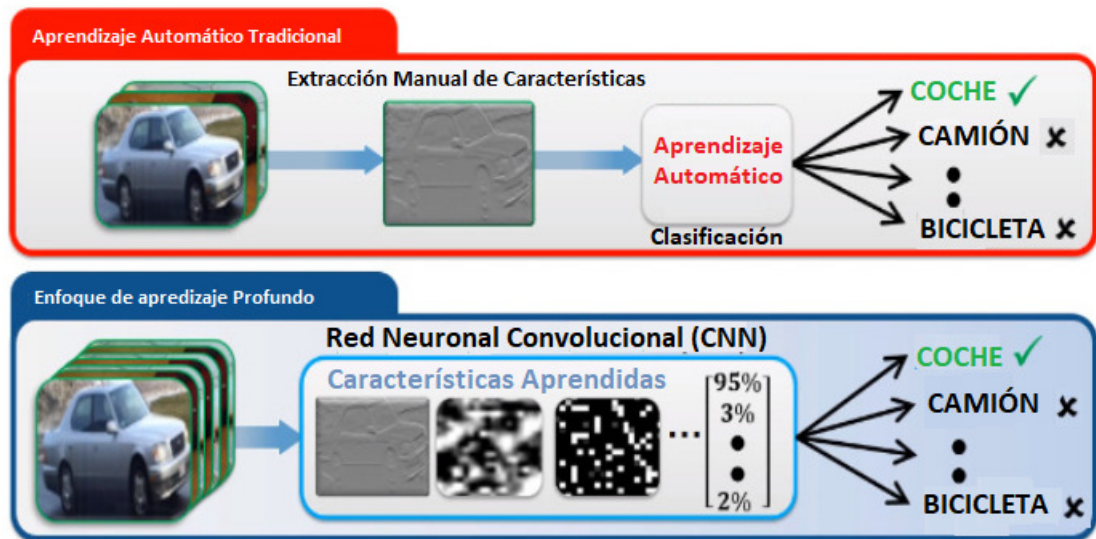


Figura 2.6: Comparación de flujo de trabajo entre aprendizaje automático y profundo

Fuente: [García, 2016]

2.2.5. Red neuronal Artificial (RNA).

Patrón computacional inspirado en la gran red de neuronas de un cerebro biológico. Recurre a un conjunto de relaciones para procesar una muestra de ingreso o registro con el fin de transformarlo en la salida deseada, pasando por varias etapas. Este enfoque se emplea con frecuencia en la clasificación de

imágenes, traductores de lenguaje y demás usos de interés actual. [DeepAI, 2019b]. Los elementos básicos de una red neuronal son:

- Conjunto de unidades de procesamiento (neuronas).
- Conexión entre unidades (asociando a cada conexión un peso o valor).
- Funciones de salida o activación para cada unidad de procesamiento.

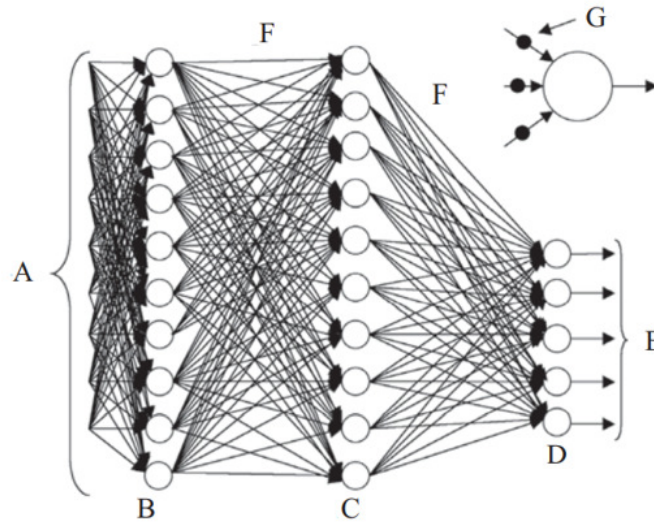


Figura 2.7: Esquema de Red Neuronal multicapas 10-10-5

Fuente: [Ponce and Herrera, 2010]

En la figura 2.7 se representa una red neuronal de diez neuronas en la capa de entrada, diez neuronas en la capa oculta y cinco neuronas en la capa de salida. Además se distinguen las entradas (A): Representa cualquier variable. La Capa de entradas (B), capa oculta (C) y capa de salidas (D). Salidas (E): Representa también cualquier variable de interés para el usuario. Los pesos entre cada neurona (F) están representados por un punto negro (G).

La clase de la neurona y la estructura de la red; detallan como se procesan sus entradas para obtener las salidas. Los componentes específicos que sirven para el computo o evaluación y que conforman la gran parte de las estructuras de modelos neurales artificiales, se los llama neuronas artificiales.

Entre las estructuras de mayor uso, teniendo en cuenta las distintas formas de efectuar sus conexiones, tenemos: a) Redes de propagación hacia

adelante (feed-forward), donde la secuencia de los datos desde las entradas hacia las salidas es únicamente hacia adelante, transmitiéndose por las diversas capas de la red, sin que haya una conexión de retroinformación. b) Redes recurrentes, están formadas por conexiones de retroinformación, lo que permitiría dirigir el flujo de la información hacia un estado invariable en el que no exista modificación en la condición de habilitación de las neuronas.[[Ponce and Herrera, 2010](#)]

Proceso de Aprendizaje o entrenamiento de una RNA, Ya se ha mencionado, que las redes neuronales artificiales toman datos de entrada; los cuales se convierten para obtener una salida, a fin de categorizarlos o afinar una función. Presumiendo que la red posea la cantidad adecuada de neuronas, será posible afinar con una determinada exactitud cualquier función continua, si se selecciona idóneamente los valores para las ponderaciones graduables en la red, también conocidos como “pesos sinápticos”; esta es la forma en que una red logra guardar la información obtenida acerca del problema a solucionar.

Dicha información se almacena por medio del proceso de aprendizaje o entrenamiento, y se refiere a los ajustes realizados en los parámetros de la RNA usando un mecanismo predeterminado, con el fin aumentar su desempeño. Igual que en el procedimiento para el aprendizaje de las personas, en las RNA este se fundamenta en la utilización de ejemplos que plantean el problema. Al colectivo de estos ejemplos se lo llama conjunto de entrenamiento. Cabe anotar que la finalidad del aprendizaje no es memorizar las interacciones existentes entre las entradas y salidas en el conjunto de entrenamiento, pero si modelar el procedimiento que ha producido estos datos. Por esto es necesario que la cantidad y clases de ejemplos usados en el entrenamiento de la red sean lo bastante representativos respecto de la relación que se desea conocer. Así, después de ser entrenada, esta podrá manejar no únicamente los ejemplos de entrenamiento, sino que también le será posible gestionar otros datos diferentes a los ya usados, sin que por esta razón se vea afectado su desempeño. A esta cualidad se la llama capacidad de generalización de la RNA.[[Palma and Marín, 2008](#)]

Se llama entrenamiento al procedimiento de estructuración de una red neuronal con el fin de que las entradas fabriquen las salidas esperadas, mediante la potenciación de sus relaciones (interconexiones). Una manera de conseguir esto es iniciar con la asignación de pesos conocidos de antemano; otra forma, es la utilización de métodos de retroinformación y esquemas de aprendizaje, los cuales alteran sus pesos hasta encontrar los más convenientes.[[Ponce and Herrera, 2010](#)]

2.2.6. Red neuronal convolucional

Es un tipo de red neuronal profunda que puede trabajar directamente con datos estructurados (como imágenes con el fin de clasificarlas); al apilar múltiples capas para formar la arquitectura de la CNN, la red es capaz de llevar a cabo el aprendizaje de las propias características, eliminando la necesidad de llevar a cabo una selección manual de las mismas; a diferencia del enfoque clásico donde se usa un número de capas reducido, utilizando CNN se utilizan un número de capas elevado posiblemente entre 5 y 50 capas incluso más, como se muestra en la fig.2.8; por este motivo, las GPU's (graphics processing unit) van a jugar un papel clave en el entrenamiento de estas redes neuronales convolucionales.

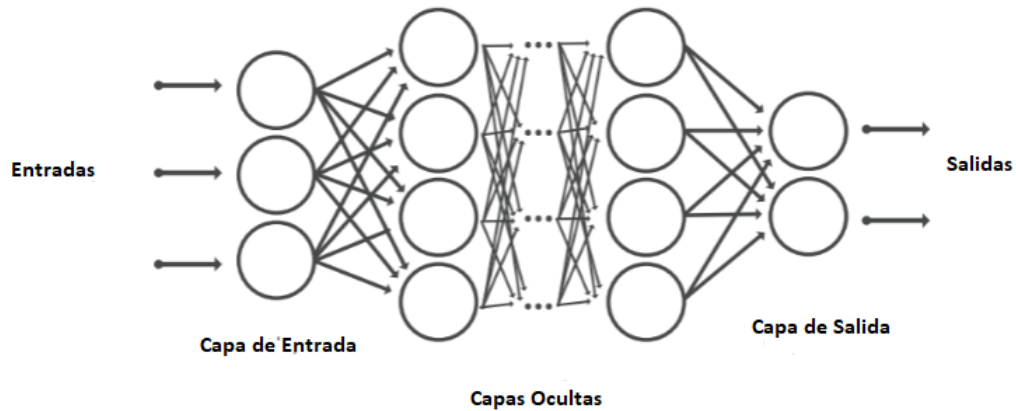


Figura 2.8: Arquitectura de una CNN

Fuente: [Mathworks, 2018]

En la figura 2.9 se representa un típico procesamiento de datos en una CNN; se puede ver a estos a medida que viajan por la red; a muy alto nivel (esto es en el primer conjunto de capas) estarán detectando características, mientras las capas finales tendrán como objetivo resolver la clasificación final de la imagen, la clave es que todas estas capas van a ser entrenadas conjuntamente, lo que quiere decir que el proceso de entrenamiento va a consistir en ajustar correctamente los pesos en todas estas capas para que las imágenes sean clasificadas correctamente. [García, 2016]

Al fijarnos con más detalle en la arquitectura de una CNN, vamos a empezar con una capa de entrada cuyos parámetros van a venir determinados por el conjunto de imágenes (en este caso imágenes de un determinado

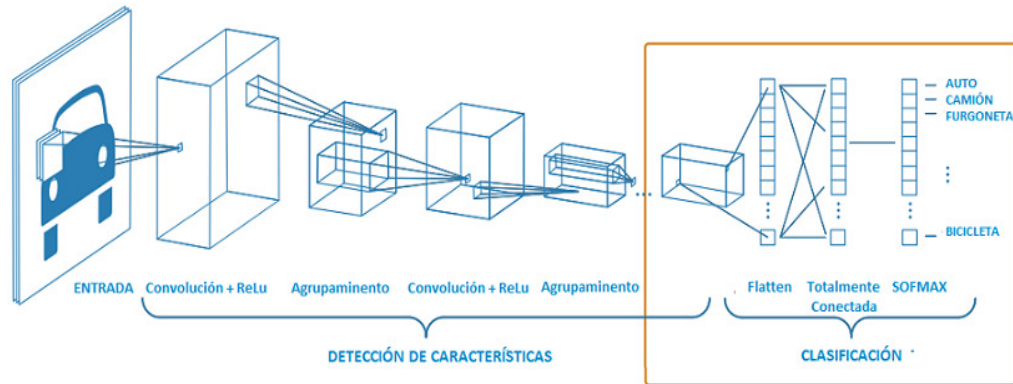


Figura 2.9: Estructura “ Capas ” CNN

Fuente: [Mathworks, 2018]

ancho y alto y 3 canales (rojo, verde y azul), a continuación podemos tener varias capas de tipo convolucion, relu y pooling (estas capas están involucradas principalmente en el aprendizaje de las características de las imágenes mediante la aplicación de convoluciones 2D). Una vez que llegamos al último volumen lo que se hace es aplanar esta capa, es decir estirarla en un vector de modo que podemos trabajarla con redes totalmente conectadas y finalmente una capa softmax que nos permite determinar a que etiqueta corresponde nuestra imagen.

Los filtros en las capas que llevan a cabo convoluciones estarán formando estructuras muy simples en capas tempranas (iniciales o primeras etapas), posiblemente segmentos verticales u horizontales, quizás pequeños bloques y estructuras más complejas a medida que nos adentramos en capas más y más profundas de modo que podemos usar estas características para distinguir diferentes objetos. [García, 2016]

La CNN es una clase de red neuronal artificial de aprendizaje supervisado, que gestiona sus capas emulando a la corteza visual del ojo humano; con el propósito de encontrar diferentes particularidades (patrones) en las entradas y así eventualmente ser capaz de clasificar correctamente objetos y “ver”. Para ese fin, la CNN está conformada por múltiples capas dedicadas ocultas, las cuales tienen una graduación determinada; en otras palabras, las capas tempranas (primeras) permiten identificar trazos rectos o arqueados, estas van tecnicándose cada vez más a medida que se vaya avanzando a capas más profundas, las cuales pueden distinguir figuras más elaboradas, como por ejemplo una cara o la forma de algún animal. [Bagnato, 2018].

Entrenar una red neuronal convolucional desde cero, este tipo de enfoque se aconseja cuando los datos disponibles para el entrenamiento consiste en miles a millones de imágenes etiquetadas para clasificar. Se debe tener en cuenta el uso de GPU's para llevar a cabo el entrenamiento de la red; ya que se trata de problemas computacionalmente muy costosos (exigentes, intensivos), pese a esto el proceso de entrenamiento puede durar días o incluso semanas para problemas reales. En cuanto a la precisión del modelo esta será muy alta pero se tendrá que tener especial cuidado cuando se utilice base de datos pequeños ya que es fácil que se produzca un sobre-ajuste. [García, 2016]. En la figura 2.10 se muestra una secuencia de los pasos requeridos para su implementación; así como el desafío que plantea cada uno de ellos.


PASOS	DESAFÍOS
Importar datos	Gestionar un gran conjunto de imágenes etiquetadas
Preprocesamiento	Cambio en el tamaño, aumento de datos
Escoger una arquitectura	Antecedentes en redes neuronales (aprendizaje profundo)
Entrenamiento	Tarea computacional intensiva (requiere GPU)
Diseño iterativo	

Figura 2.10: Pasos y Desafíos para entrenar una red convolucional desde cero
Fuente: [García, 2016]

Afinar una Red Neuronal pre-entrenada, consiste en usar una red convolucional ya entrenada y utilizarla para un nuevo propósito, clasificar un conjunto de imágenes diferentes. De este modo utilizaremos una CNN que ha sido entrenada para clasificar un conjunto de imágenes muy grande y donde ya se han aprendido un conjunto de características, lo que se hace es ajustar los pesos de las últimas capas mediante un re-entrenamiento de modo que se pueda clasificar nuevos conjuntos de datos. Este flujo de trabajo está recomendado para un conjunto de datos menor siendo su entrenamiento menos intenso computacionalmente. Evidentemente la precisión del modelo va a estar directamente relacionada con la calidad de la red pre-entrenada. En la figura 2.11 muestra este enfoque, como ejemplo se ajusta un modelo previamente entrenado (aprendizaje por transferencia); utilizando la red Alex-net, para categorizar los vehículos en dos clases como auto o SUV.

Existen varias redes neuronales preentrenadas que nos pueden ayudar



Figura 2.11: Proceso Transfer Learning para la identificación de Objetos
Fuente: [García, 2016]

a realizar una clasificación de manera más sencilla tan solo volviéndola a entrenar con los parámetros que mejor se adapten a la selección del problema.

A continuación se dan detalles de las redes preentrenadas usadas por el software utilizado en el presente proyecto.

ResNet.- Empleada de manera particular como el componente fundamental en gran cantidad de actividades de visión artificial. Fue la red triunfadora del concurso ImageNet del 2015. Su avance más notable tiene que ver, con que permitió al fin entrenar redes muy profundas (con un número de capas mayor a 100); manejando con gran acierto el inconveniente del desvanecimiento de gradiente.

En la figura 2.12 se realiza la comparación de las arquitecturas de tres redes distintas, estas son: Estructura VGG19, red convolucional con 19 capas ganadora del ImageNet año 2014 mostrando desaciertos en el 7.4% de las imágenes (lado izquierdo). Como referencia (en el centro) se describe una arquitectura convolucional de 34 capas de parámetros (152, sin ligas puentes) y finalmente una estructura ResNet, red residual con 34 capas de parámetros (152) triunfadora en la misma competición en el 2015; observándose tan solo el 3.6% de desaciertos (situada a la derecha).

Los desarrolladores de la red residual dieron a conocer 2 variantes de la estructura ResNet; en la posterior versión optimizaron el rendimiento de la primera. Siendo la segunda versión la más sofisticada e intuitiva.

El fundamento de la ResNet, es incrementar el rendimiento de las redes conocidas como totalmente convolucionales (Fully Convolutional Networks, FCN). En un principio estas fueron desarrolladas con el fin de evaluar una imagen (parche) en base a una imagen (o parche); en contraste a las redes convencionales que únicamente examinan el valor de un píxel, ya presentada la proximidad del mismo (parche). Mediante este enfoque se tramitan la totalidad de los píxeles del parche con un único paso [Rivera, 2019].

Los módulos o bloques residuales son los elementos fundamentales en la red ResNet. En el módulo residual, la entrada " \mathbf{x} " se añade inmediatamente a la salida de la red; cumpliéndose de esta manera con: « $\mathbf{F}(\mathbf{x}) + \mathbf{x}$ »; a este trayecto se le llama “conexión de salto” o “de acceso directo” [He et al., 2016]. En la figura 2.13 se representa el esquema de un bloque residual básico.

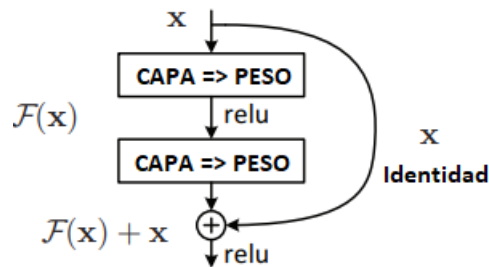


Figura 2.13: Diagrama de un bloque residual

Fuente: [He et al., 2016]

Ya examinado como se forma un bloque residual, lo que hacemos para diseñar una red residual, es agregar múltiples bloques residuales dentro de su arquitectura. En base a esta premisa, los desarrolladores han estructurado muchas posibilidades de redes residuales con distintos números de capas como ResNet34 y ResNet50. En la figura 2.12 se observa una ResNet34 (red residual de 34 capas con conexión de salto).

MobileNet.- Es una modalidad de redes neuronales convolucionales, diseñada especialmente para aplicaciones de visión computacional en dispositivos móviles e integrados. Se fundamentan en una arquitectura optimizada que utiliza convoluciones separables en profundidad (un tipo de convolución factorizada), para construir redes neuronales profundas con un peso ligero. A cada canal de entrada se aplica un único filtro por medio de

la convolución en profundidad y una convolución 1×1 llamada convolución puntual para combinar los resultados. Una convolución estándar filtra y combina las entradas en un nuevo conjunto de salidas en un solo paso. La convolución separable en profundidad divide esto en dos capas, una capa separada para filtrar y una capa para la combinación. Esta factorización tiene el efecto de reducir drásticamente el cálculo y el tamaño del modelo. [Howard et al., 2017] Las diferencias existente con los filtros separables en profundidad y los estándares se aprecia en la figura 2.14.

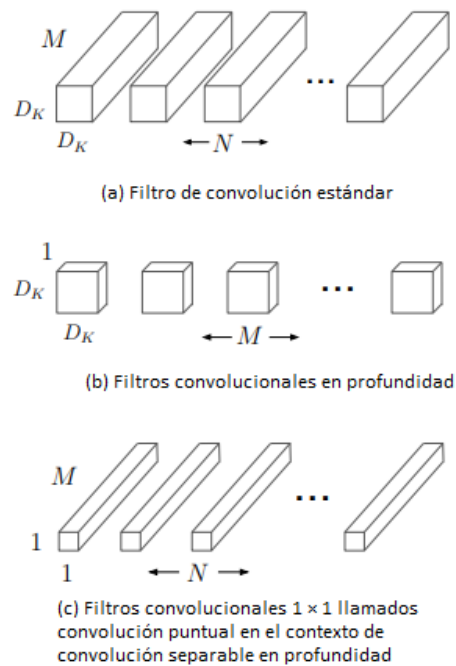


Figura 2.14: Filtros separables en profundidad

Fuente: [Howard et al., 2017]

Los filtros convolucionales estándar en (a) se reemplazan por dos capas: una de convolución en profundidad en (b) y otra de convolución puntual en (c) para construir un filtro separable en profundidad

Así, la estructura básica de las MobileNets además de un bloque de estandarización y la función de activación ReLU correspondiente es visualizada del lado derecho en la Figura 2.15, mientras que a la izquierda esta una estructura capa capa convolucional estándar con módulo de normalización y ReLU

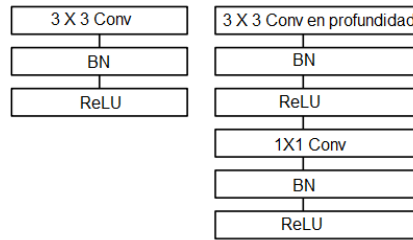


Figura 2.15: Comparación de Estructuras tipo estándar con Mobilenet
Fuente: [Howard et al., 2017]

Los autores de las MobileNets presentaron en el 2018, una nueva versión, conocida como MobileNet V2 [Sandler et al., 2018]. En este desarrollo se incorpora nuevos planteamientos, entre ellos tenemos el de residuos invertidos y cuellos de botella lineales; una propiedad interesante de esta arquitectura es que proporciona una separación natural entre los dominios de entrada/salida de los bloques de construcción (capas de cuello de botella), y la transformación de las capas; es decir, una función no lineal que convierte la entrada en la salida. En nuestro caso en particular, cuando la profundidad de la capa interna es 0, la convolución subyacente es la función de identidad gracias a la conexión de atajo. Cuando la expansión es menor que 1, se trata de un bloque convolucional residual clásico. La Figura 2.16 muestra el bloque convolucional básico de MobileNet V2.

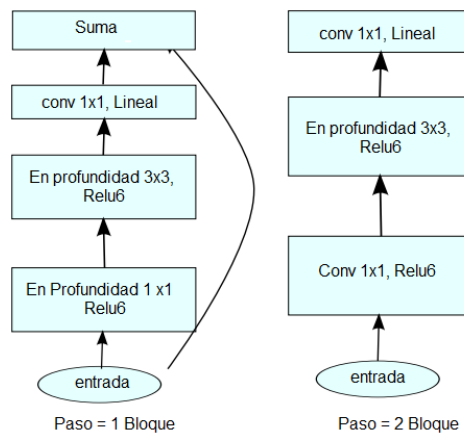


Figura 2.16: Bloque básico Mobilenet V2
Fuente: [Sandler et al., 2018]

Lobe: Herramienta-aplicación informática desarrollada por Microsoft que permite construir, entrenar y reentrenar modelos de aprendizaje automático personalizados; que se pueden exportar en una variedad de formatos, para luego ser desplegados en cualquier plataforma que se desee. Lobe posibilita crear modelos de aprendizaje automático utilizando una interfaz visual simple en lugar de escribir líneas de código para su programación. En la versión actual se puede clasificar conjuntos de imágenes para que Lobe pueda identificar el contenido de imágenes similares. Es una alternativa de las herramientas “low-code para AI”. Cuyo propósito es entrenar modelos de deep learning de manera simple para que después se puedan integrar en una app.[Cabot, 2018]

De manera general podemos detallar la siguiente secuencia para la obtención del modelo.

- Colectar e identificar las imágenes.
- Entrenar el modelo
- Probarlo
- Determinar o detectar debilidades y/o errores
- Tratar de mejorarlo/corregir errores o malas predicciones
- Reentrenar

Lobe elige la mejor arquitectura del modelo de aprendizaje automático para la clasificación de imágenes entre dos opciones:

- **Precisión:** ResNet-50 V2.
- **Velocidad:** MobileNetV2.

Ambos modelos utilizan el aprendizaje por transferencia con pesos previamente entrenados del conjunto de datos de ImageNet. La transferencia de aprendizaje le permite entrenar mejores modelos con menos datos y ofrece un mejor punto de partida para entrenar con datos más grandes.

Tensor flow: Es una biblioteca de código abierto para el desarrollo e implementación modelos de aprendizaje profundo, usa gráficos computacionales para el flujo de datos y cálculos numéricos. En otras palabras, los datos, o tensores, fluyen a través del gráfico, de ahí el nombre tensorflow. El gráfico tiene nodos que permiten cualquier cálculo numérico y, por lo tanto, son adecuado para operaciones de aprendizaje profundo. Proporciona una única API (Application Programming Interface: Interfaz de programación de aplicaciones) para todo tipo de plataformas y hardware.

TensorFlow maneja toda la complejidad del escalado y la optimización en el backend (parte del desarrollo web que se encarga de que toda la lógica de una página web funcione). Originalmente fue desarrollado para investigación en Google. Esta es la biblioteca de aprendizaje profundo más famosa, con una gran comunidad y viene con herramientas para visualización y despliegue en producción[Shanmugamani, 2018].

Métricas de evaluación de clasificación.- Son medidas del rendimiento de un clasificador, básicamente un porcentaje del acierto o error sobre un conjunto de prueba. Por lo tanto lo primero que necesitamos para poder evaluarlo es el conjunto de imágenes de prueba también llamado conjunto de evaluación, cada imagen en el conjunto de evaluación debe estar etiquetada con la categoría que esperamos que se devuelva como resultado de la clasificación, este etiquetaje con su resultado esperado se conoce como Ground truth y es indispensable en cualquier tarea de evaluación de rendimiento. Previo a la definición de estas métricas debemos conocer los siguientes conceptos:

- **Matriz de confusión o error:** Es una tabla que nos permite representar el comportamiento del clasificador sobre cada clase. Describe el rendimiento de un modelo supervisado de Machine Learning en los datos de prueba. Se llama “matriz de confusión” porque hace que sea fácil detectar donde el sistema está confundiendo dos clases.

En las filas de la matriz se representan las imágenes del conjunto de evaluación; es decir, en cada fila vamos a analizar las imágenes de una de las categorías que tenemos definidas en el Ground Truth. Por otro lado en las columnas, se van a representar el resultado de la clasificación de cada categoría. Por lo tanto en las intersecciones entre filas y columnas se puede analizar el resultado del clasificador para cada una de las categorías del Ground Truth utilizado, en la figura 2.17 se representa una matriz de confusión binaria (solo se determina

si pertenece o no a la categoría pre-establecida); aquí a los datos de prueba se le da el valor de 1 si pertenece a la categoría (cumple condición que se desea predecir) y en caso contrario toma el valor de 0.

Matriz de Confusión

Clase de Salida	0	65 31.9%	23 11.3%	73.9% 26.1%
	1	20 9.8%	96 47.1%	82.8% 17.2%
		76.5% 23.5%	80.7% 19.3%	78.9% 21.1%
		0	1	
		Clase Objetivo		

Figura 2.17: Ejemplo de Matriz de confusión binaria

Fuente: el autor

Una vez hecha la matriz de confusión podemos definir los siguiente subconjuntos para un clasificador binario:

- Verdaderos Positivos (VP o TP): Cuando la clase real de la muestra de datos observada coincide con la categoría clasificada. Cabe señalar que para un clasificador multiclase estos valores se encuentran en la diagonal principal de la matriz de confusión.
- Verdaderos Negativos (VN o TN): Predicciones donde la muestra clasificada se señala como no incluida en una clase, acertando en su categorización.
- Falsos Positivos (FP): Se dan cuando a pesar de que la muestra o entrada no pertenece a una clase, el predictor indica que si.
- Falsos Negativos (FN): Predicciones donde el dato a clasificar se identifica como si no formará parte de una clase, a pesar de que sí pertenece a esta.

Ya definidos estos conceptos podemos seguir con las métricas.

- Exactitud.- Es el porcentaje total de elementos pertenecientes al conjunto de prueba que fueron clasificados correctamente, en la ecuación 2.1 se muestra la formula para su calculo en un clasificador binario

$$Exactitud = \frac{VP + VN}{VP + FP + VN + FN} \tag{2.1}$$

Nota: En el caso de un clasificador multiclase no se tiene el concepto de verdaderos negativos (VN) ya que todas las imágenes van a quedar clasificadas en una de las categorías, en la figura 2.18 se muestra un ejemplo de una matriz de confusión de un clasificador para cuatro clases distintas (Persona, Coche, silla y Gato); teniendo en cuenta lo anterior la ecuación 2.2 calcula la exactitud para un clasificador multiclase.

$$Exactitud = \frac{VP}{No. Total de Predicciones} \tag{2.2}$$

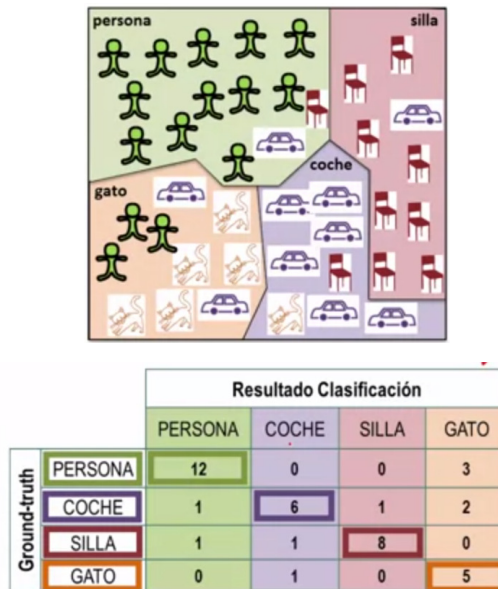


Figura 2.18: Matriz de Confusion Clasificador Multiclase

Fuente: [Valveny]

- **Precisión.**- Es la porción de clasificaciones de valor afirmativo que son verdaderas; en otras palabras, la fracción de verdaderos positivos sobre el total de datos del conjunto de prueba que son clasificados como positivos; así tenemos la ecuación 2.3

$$Precisión = \frac{VP}{VP + FP} \quad (2.3)$$

- **Exhaustividad o sensibilidad (Recall o VPR).**- Es una medida de la cualificación correcta de casos positivos; es decir, la porción de las entradas que corresponden a la clase de interés que se identificaron como tales; en otras palabras, es la cantidad de datos categorizados acertadamente como positivos del total de positivos reales. Su calculo es en base a la ecuación 2.4.

$$Sensibilidad = \frac{VP}{VP + FN} \quad (2.4)$$

- **Especificidad o TNR (Tasa Negativa Verdadera).**- Porcentaje de la cantidad de muestras o datos acertadamente reconocidos como negativos del total de negativos. La ecuación 2.5 calcula su valor.

$$Especificidad = \frac{VN}{VN + FP} \quad (2.5)$$

- **F1-Score.**- Esta medida mezcla dos tasas previas (precisión y sensibilidad); para, en un solo indicador dar el rendimiento de las predicciones positivas. Se define como una media armónica que combina las métricas ya señaladas. Siendo calculada mediante la ecuación 2.6

$$F1 = 2 * \frac{Precisión * Sensibilidad}{Precisión + Sensibilidad} \quad (2.6)$$

- **AUC (Área bajo la curva de funcionamiento del receptor).**- Con el fin de conceptualizar este indicador; en primer lugar hay que referirse a la curva característica operativa del receptor (ROC), este es un gráfico que posibilita valorar el desempeño de un identificador de clases en función de su sensibilidad versus su especificidad, tomando en cuenta todos los valores de umbral posibles; en otras palabras refleja con distintos umbrales de clasificación, los resultados de la sensibilidad

contra la tasa de falsos positivos FPR (1-Especificidad), en la figura 2.19 se muestra un ejemplo de curva ROC.

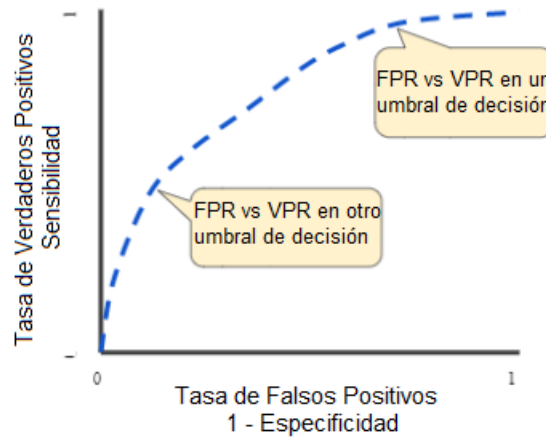


Figura 2.19: Gráfica ROC

Fuente: [Google-Developers, 2019]

La medida de rendimiento obtenida de las ROC es el área bajo la curva (AUC); a esta se la puede definir como el aproximado de la probabilidad, de que un clasificador otorgue un valor más alto a una muestra positiva seleccionada de manera aleatoria, que a una negativa escogida al azar. Consta de un algoritmo fundamentado en ordenamiento para estimar el área bajo la curva ROC entre las coordenadas (0,0) y (1,1).

Por lo tanto el valor de AUC oscila entre 0 y 1, un valor de cero supone un modelo que siempre se equivoca en sus predicciones; mientras si el valor de AUC es uno significa que todas sus predicciones son correctas. Entre los beneficios de emplear el área bajo la curva característica operativa del receptor, están que es invariable en cuanto a la escala e invariante del lugar de clasificación. [Google-Developers, 2019]. AUC es invariante en escala porque mide que tan bien se clasifican las predicciones, en lugar de sus valores absolutos. AUC es de clasificación-umbral-invariante es decir mide la calidad de las predicciones del modelo independientemente del umbral de clasificación elegido.

Por ejemplo dados los datos, que se muestran en la figura 2.20 donde estos organizan de izquierda a derecha en orden ascendente de predicciones. AUC representa la probabilidad de que un ejemplo positivo aleatorio (verde) se coloque a la derecha de un ejemplo negativo aleatorio (rojo).

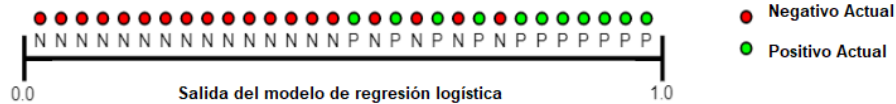


Figura 2.20: Predicciones clasificadas en orden ascendente de puntuación

Fuente: [Google-Developers, 2019]

AUC Multiclase. Como ya se ha mencionado el AUC se evalúa sobre dos clases. En los casos de clasificación binaria este valor es un simple número, sin embargo en el caso de múltiples clases (más de dos); se presenta la tarea de conjuntar todos los valores de AUC obtenidos de cada par de clases. Su cálculo se basa en el criterio de que AUC es igual a la probabilidad que un clasificador confiera un valor más alto a un ejemplo positivo seleccionado al azar que a uno negativo determinado de manera aleatoria. [Hand and Till, 2001]. Así se da la ecuación 2.7.

$$AUC = \frac{2}{|C|(|C| - 1)} * \sum_{c_i, c_j \in C} AUC_{(c_i, c_j)} \quad (2.7)$$

Aquí C representa la cantidad total de las clases y $AUC_{(c_i, c_j)}$ describe el área bajo la curva ROC que asocia la clase c_i y c_j .

Otro criterio que permite obtener el AUC multiclase, es el cálculo del AUC esperado o media ponderada de los AUC; para esto, se debe generar la curva ROC de cada clase, tomándola a su turno como referencia; es decir, convirtiéndola en la clase 0 y a todas las demás clases tomarlas como clase 1, para así encontrar su respectivo AUC. Luego realizar la sumatoria de las AUC sopesadas por su predominancia en los datos [Provost and Domingos, 2000]. Si $p(c_i)$ es el peso del AUC de la clase i (la frecuencia de la clase i en los datos), obtenemos la ecuación 2.8.

$$AUC_{total} = \sum_{c_i \in C} AUC(c_i)p(c_i) \quad (2.8)$$

2.3. Formulación del Problema

Las Naciones Unidas en su informe del 2016 “Agua y Empleo”, estima que cerca del 80 % de las plazas de empleo del mercado laboral global, requiere de la disponibilidad en el abastecimiento apropiado del agua y de los servicios asociados con esta. Debido a que la competencia por la provisión de agua dulce se incrementa, y que el calentamiento global incide de manera negativa en sus suministros, se hace cada vez más prioritario que los países elaboren e implementen reglas de ocupación laboral que consideren las restricciones establecidas por la asequibilidad de este recurso. No perjudicar la posibilidad del sostén de las fuentes de agua y los ecosistemas es fundamental en aras de garantizar la sustentabilidad colectiva, monetaria y medio-ambiental a largo plazo [[Organización de las Naciones Unidas para la Educación, 2016](#)].

Según el Instituto Nacional de Estadística y Censos, El Ecuador posee una población de aproximadamente 17,3 millones de habitantes a Julio del 2019, [[INEC-Censos](#)]; contabilizándose que un 77 % de las familias se deshace de su basura por medio de los carros recolectores y el 23 % restante la elimina de distintas maneras; como por ejemplo, “la arroja a terrenos baldíos o quebradas, la quema, la entierra, la deposita en ríos acequias o canales, etc.” [[Ministerio de Ambiente](#)]

La acumulación de residuos o desechos en ambientes acuáticos que son resultados de las actividades humanas (entre las cuales podemos destacar las industriales, agrícolas, domésticas con sus desechos domiciliarios, etc) es creciente y cada vez más compleja; éstas, durante las últimas décadas han puesto en serio riesgo al ambiente y consecuentemente a la salud humana.

Para que el modelo de sociedad industrializada en que vivimos pueda mantenerse y no afectar de manera crucial nuestra la calidad de vida, debe cuidarse de manera eficaz este recurso natural; que cada vez se vuelve más escaso. Supervisar los niveles de contaminación y monitorear de manera continua los espejos de fuentes de agua dulce debe ser prioridad para nuestra sociedad.

La ausencia de mecanismos y herramientas que puedan determinar casi de manera instantánea los objetos que se encuentran en la superficie de los ríos, lagos, mares, etc. impide una rápida actuación de los entidades involucradas en mantener su calidad y cuidado.

Recientemente la concienciación del cuidado al medio ambiente es una prioridad en la investigación, innovación y desarrollo; el encontrar soluciones ingeniosas, de rápida implementación y bajo costo es fundamental para poder hacer frente a los desafíos que actualmente presentan la contaminación de afluentes de agua dulce.

Capítulo 3

Marco Metodológico

En este capítulo se expone minuciosamente el procedimiento llevado a cabo en el desarrollo del sistema de identificación; se menciona entre otros, los aspectos de selección de elementos y equipos usados en la implementación del prototipo para solucionar el problema planteado. Así como los tipos de software y códigos empleados. Además, se discute cualitativamente los resultados obtenidos y se plantean posibles mejoras para solventar las dificultades encontradas.

3.1. Diseño del experimento

Podemos considerar que el experimento consiste en cuatro etapas, las cuales se detallan a continuación:

- Obtención de las muestras de agua con los diferentes agentes contaminantes seleccionados (captación de imágenes). Previo a esto se definió las categorías que se van a modelar como siguen: Agua **Potable**, la cual se representa con una muestra de agua limpia sin más componentes que agua pura; con **Sólidos**, que simula los desechos orgánicos; **Aceite**, para imitar los desechos industriales con señas de hidrocarburos; **Turbia**, que representa polución, suciedad, sedimentación, etc.
- Entrenamiento del modelo de aprendizaje automático utilizando Lobe: familiarización del entorno, uso, funciones y operación de esta herramienta.
- Pruebas del modelo, generando y utilizando nuevas muestras (no empeladas en el proceso de entrenamiento). En este punto se utiliza el modelo obtenido para realizar la comparación y clasificación de un conjunto de imágenes diferentes a las ya usadas durante el proceso de entrenamiento.
- Implementación del dispositivo clasificador, que entregue una respuesta de la comparación e identificación de la imagen (muestra); en forma de una señal digital como resultado para cada una de las categorías señaladas.

Elementos y Dispositivos seleccionados para el desarrollo.

Para realizar la captación de imágenes se seleccionó la tarjeta de desarrollo **Raspberry Pi 4 de 4 GB**; debido a sus prestaciones tecnológicas, tanto para la recolección de datos como para su procesamiento, ya que tiene la capacidad de adaptar una cámara y posee una CPU ARM Cortex-A72 la cual permite entre otras cosas, la decodificación de vídeo 4K a 60 fps (sin compatibilidad con HDR); además, de su disponibilidad, costo y facilidad de programación. Este hardware en particular con su nueva versión de procesador es capaz de soportar las librerías de lobe y tensor flow necesarias para poder desarrollar la programación del clasificador. En la fig 3.1 se muestra la vista superior del dispositivo describiendo la ubicación de los diferentes puertos de conexión para sus interfases.

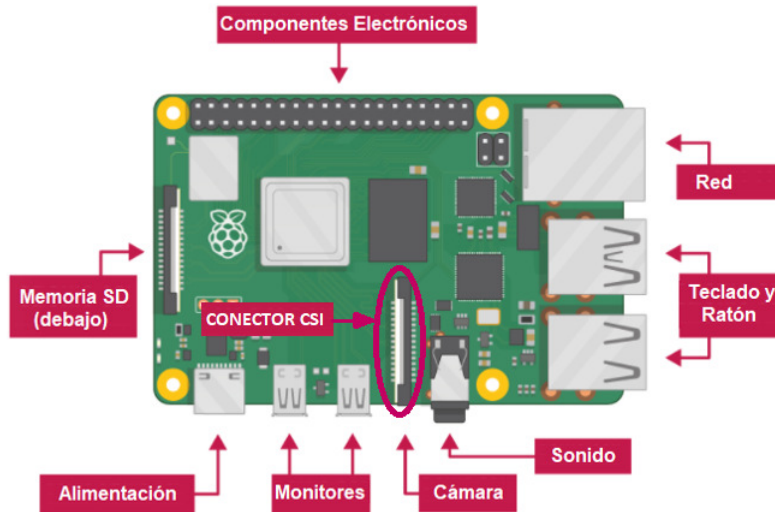


Figura 3.1: Raspberry Pi 4, descripción y ubicación de sockets e interfaces
Fuente: [Raspberrypi-Projects, 2017]

Características de la cámara V 1.3: El módulo de la cámara para Raspberry Pi se conecta directamente al conector CSI de la tarjeta, el cual se muestra en la figura 3.1. Permite proporcionar una captura de imagen nítida con una resolución de 5MP o una filmación de video en alta definición de 1080p a 30 fps.

El módulo de cámara de la Raspberry Pi, dispone de un detector Omnivision 5647 de 5MP (2592 X 1944 píxeles) en una unidad con orientación fija. El dispositivo se conecta a la Raspberry Pi, por medio de un cable plano de 15 pines, a la interfaz en serie dedicada a cámara (CSI) de 15 pines, que fue diseñada especialmente para interconectarse con cámaras. El bus CSI es capaz de soportar velocidades de datos extremadamente altas y transporta exclusivamente datos de píxeles al procesador. La placa en sí es pequeña, de alrededor de 25 mm x 20 mm x 9 mm y pesa poco más de 3 g, lo que la hace perfecta para aplicaciones móviles o de otro tipo donde el tamaño y el peso son aspectos importantes. En términos de imágenes fijas, la cámara es capaz de capturar imágenes estáticas de 2592 x 1944 píxeles y también admite grabación de video 1080p @ 30fps, 720p @ 60fps y 640x480p @ 60/90. La cámara es compatible con la última versión de Raspbian, el sistema operativo más usado en la Raspberry Pi. Esta cámara está diseñada para capturar imágenes con luz natural. [Industries-Adafruit]

3.2. Desarrollo

Inicialización y Configuración de la Raspberry Pi: Para este procedimiento necesitaremos los siguiente elementos y accesorios:

- Una fuente de alimentación de 5.1 Vcc - 3A ,con conector C para USB.
- Una tarjeta de memoria Micro SD: Esta sirve para almacenar todos los archivos y el sistema operativo Raspberry Pi OS, en nuestro caso usamos una de 16 GB de capacidad.
- Un cable convertidor de HDMI a micro HDMI, ya que por ser una Raspberry Pi 4 solo posee 2 puertos micro HDMI, uno de estos es usado para conectar la pantalla.
- Una pantalla de TV o computador, en nuestro caso se usa un monitor de PC.
- Un teclado y un mouse ambos cableados (una vez ya configurado la Raspberry Pi, se puede usar un teclado y un mouse Bluetooth).

La Raspberry Pi necesita un sistema operativo para funcionar. El Raspberry Pi OS (anteriormente llamado Raspbian) es el sistema operativo compatible oficial.

Se usó la herramienta **Raspberry Pi Imager** (desarrollada por Raspberry); la cual puede ser descargada en una PC o ejecutada directamente desde la web (<https://www.raspberrypi.org/software/>) para instalar el sistema operativo Raspberry Pi OS (32-bit) en la tarjeta SD, realizando los siguientes pasos:

- Se Insertó la tarjeta al lector de micro SD de la PC, en nuestro caso se usó un convertidor de Micro SD a SD.
- Se seleccionó el sistema operativo Raspberry Pi OS (32-bit) de entre las opciones que aparecen al presionar **“CHOOSE OS”**.
- Se escogió la MSD como elemento de almacenamiento del sistema operativo al presionar **“CHOOSE STORAGE”**
- Finalmente dando click en la opción **“WRITE”** y Verificando la Micro SD como elemento para la descarga, se inició el proceso de instalación del sistema, se debe tener en cuenta que durante este paso todo el contenido previo de la Micro SD se perderá ya que será sobre escrita.

Ya ejecutado el flash en la tarjeta Micro SD, se está en capacidad de usar la Raspberry Pi. En la figura 3.2 se muestra el entorno del Pi Imager.



Figura 3.2: Pantalla Pi Imager

Fuente: [Foundation-Raspberry]

Configuración y Uso de la Cámara

- **Conexión.-** Se tira verticalmente de los extremos del sujetador plástico del conector y se inserta el bus de datos; de tal manera que la marca azul quede frente a los puertos USB, como se ve en la figura 3.3

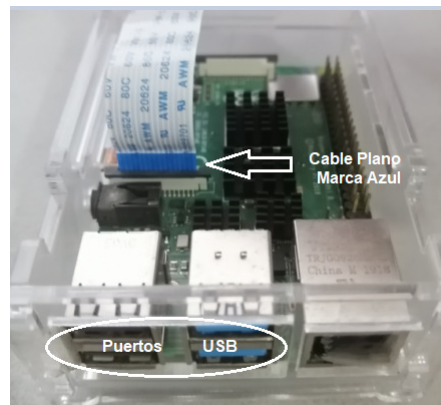


Figura 3.3: Conexión correcta cable plano

Fuente: el autor

- **Configuración.-** Encender la Raspberry Pi ya inicializada y en el menú principal se abre el entorno de configuración, se selecciona la opción “**Interfaces**” y se habilita la cámara, esta secuencia se describe en la figura 3.4, finalmente reiniciar la Raspberry Pi.

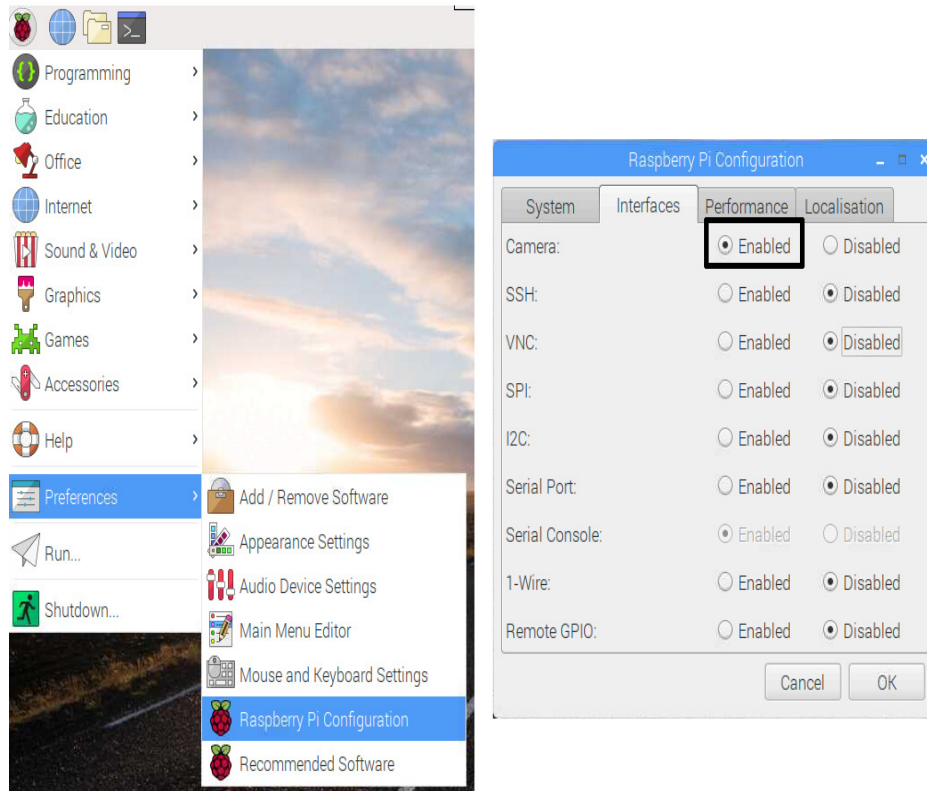


Figura 3.4: Configuración Cámara

Fuente: [raspberrypi.org, 2017]

Con el dispositivo de cámara ya conectado y su software activado, la cámara se puede controlar a través de la línea de comando, usando las órdenes **raspistill** y **raspivid** (para tomar fotos y grabar videos respectivamente). A modo de prueba se abrió la ventana del terminal (pulsando con el mouse sobre el icono de la pantalla negra que contiene los símbolos de un prompt, ubicado en la barra de tareas); ya en este entorno, se usa la orden **raspistill** para capturar una foto fija y almacenarla en formato jpg en el escritorio, digitando la siguiente línea de instrucción: **raspistill -o Escritorio / image.jpg**; se ejecuta el

comando presionando enter y a continuación se inicia la vista previa de la cámara y queda activa por cinco segundos previo a la captura de la imagen. El formato del archivo, así como la dirección donde se lo va a guardar; pueden escogerse a voluntad cambiando el respectivo argumento.

Nota: Una foto tomada con el módulo de la cámara tendrá un tamaño aproximado de 2,4 MB; implica 425 fotos por GB. Tomar 1 foto por minuto ocuparía 1 GB en aproximadamente 7 horas. Esta es una tasa de aproximadamente 144 MB por hora o 3,3 GB por día.

Recolección de Muestras

Para realizar las capturas de las imágenes necesitamos producir muestras de agua que contengan los diferentes contaminantes. Con el fin de crear un ambiente controlado; en cual, se puedan generar diferentes patrones para cada uno de los agentes contaminantes seleccionados, se escogió (como mejor alternativa de las evaluadas) un recipiente de cristal transparente de forma rectangular para contener el líquido, sus dimensiones útiles son 35X23X7 cm (largo, ancho y profundidad respectivamente).

Además se representó los agentes contaminantes usando diferentes compuestos en el agua; como se detalla a continuación:

- Agua *Potable*, muestra sin ningún elemento adicional solo agua limpia.
- Agua con *Sólidos*, usando virutas y trozos de madera pequeños
- Agua con *Aceite*, para generar este tipo de muestra se derrama achiote sobre la superficie.
- Agua *Turbia*, a fin de crear este efecto en la muestra se usa tinta de origen vegetal de color negro.

A fin de poder maniobrar la Raspberry Pi con la cámara ya conectada, se montó ambos elementos en una carcasa plástica(case). Luego se la ubicó de tal manera que quede por encima de la muestra, para esto se utilizó una mesa plegable pequeña, a la cual se le ajustó su altura; para que se enfoque la mayor cantidad de superficie de la muestra sin los bordes (solo se capte el fondo).



Figura 3.5: Estación de Muestreo montada

Fuente: el autor

Además para de evitar crear confusiones al modelo de predicción, por el cambio en el tipo de fondo (color y forma) al ser llevada la estación de un lugar a otro, se crea un fondo blanco usando un pliego de fomix por debajo del recipiente que contiene la muestra, en la figura 3.5 se observa como quedó la estación de colección de muestras ya montada.

Se creó la base de datos a usar (imágenes); tomando 130 fotos para cada clase de contaminante creada, teniendo en cuenta las siguientes consideraciones:

- **Capturar tantas “variaciones” como sea posible;** todas las variaciones que ocurrirían naturalmente se intentan simular mediante la recopilación de imágenes en diferentes condiciones iluminación, orientaciones o zoom. Esto ayuda a Lobe a saber qué partes de la imagen son útiles para hacer predicciones y qué es ruido.
- **Asegurarse de que su contenido sea visible en el cuadrado central de la imagen,** ya que mientras se entrena el modelo, Lobe recorta el cuadrado central de las imágenes.

Formas de realizar la captura de Imágenes

- **Desde la ventana terminal a través de comandos**, a continuación mencionamos algunos comandos útiles.

raspistill -o : Toma una foto fija con una resolución por default de 2.920 X 1.944 es decir de 5'038.848 o 5 megapíxeles; se debe agregar como argumentos el nombre y formato de la captura (extensión del archivo); además se tiene indicar la dirección donde se desea guardar la imagen. Junto a este comando podemos usar las siguientes opciones:

-vf y **-hf** : Son las opciones para volteo vertical y giro horizontal La imagen puede girarse 180 ° para que se muestre correctamente.

-rot : Establece la rotación de la imagen entre (0 - 359°), este puede tomar cualquier valor desde 0 hacia arriba, pero debido a restricciones de hardware, solo se admiten rotaciones de 0, 90, 180 y 270 °.

-h y **-w** : Se usan para cambiar la altura y el ancho de la imagen respectivamente; por ejemplo una imagen con ancho de 640 y alto de 480 pixeles, se puede ejecutar desde la ventana terminal de la siguiente manera: *raspistill -o Escritorio / image-small.jpg -w 640 -h 480*.

-p y **-f** : Ambos permiten al usuario definir la configuración de la ventana de vista previa, con **-p** se puede definir el tamaño y su ubicación en la pantalla con los argumentos <'x,y,w,h'>. Mientras que la vista de pantalla completa (utiliza toda la pantalla), se logra utilizando con la opción **-f**. Si se desea deshabilitar la ventana de vista previa por completo se usa **-n**.

-sh, **-co** y **-sa** : Se usan para establecer respectivamente la nitidez, contraste y saturación de la imagen en un rango de (-100 - 100); siendo 0 el valor predeterminado.

-br : Establece el brillo de la imagen entre (0 - 100); siendo 50 el valor predeterminado. 0 es negro y 100 es blanco.

-op : Establece la opacidad de la ventana de vista previa considerando el siguiente rango (0 - 255). Donde 0 = invisible y 255 = completamente opaco.

-ISO : Configura el ISO a ser usado en la captura, (ISO es la sensibilidad del sensor a la hora de captar la luz. A mayor número de ISO, mayor capacidad para captar luz; a menor valor, menor capacidad para capturar dicha luz). Su rango está entre (100 - 800)

-ev : Ajusta la compensación EV de la imagen entre (-10 - 10). Su valor predeterminado es 0. EV es el valor de exposición (en inglés exposure value o EV) es una combinación de diafragma (el número f), velocidad de obturación e ISO. Indica una determinada cantidad de luz que llega al sensor.

-ex : Establece el modo de exposición, Las posibles opciones son: automático, noche, vista previa nocturna, luz de fondo (selecciona la configuración para el sujeto a contraluz), destacar (spotlight), deportes (obturador rápido, etc.), nieve (snow), playa, muy largo (ajuste para exposiciones prolongadas), Fixed fps (restringe los fps a un valor fijo), antishake (modo anti sacudidas) y fuegos artificiales

- **Control del módulo de la cámara con código en Python**, utilizando el editor de Python 3 *Thonny Python IDE* desde la raspberry se puede abrir, crear, guardar y ejecutar un archivo (programa); en el cual se pueda realizar la rutina de ejecución del módulo de la cámara. La biblioteca *picamera* de Python posibilita comandar el dispositivo de la cámara y desarrollar proyectos con una estructura sistematizada. En la figura 3.6 se muestra ubicación del editor Thonny Python.

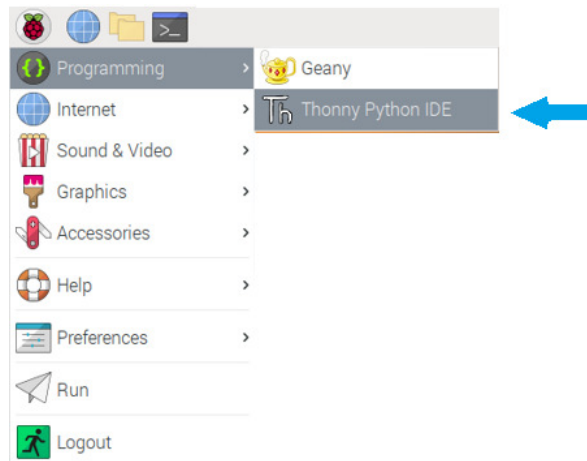


Figura 3.6: Ubicación Editor Thonny Python IDE

Fuente: el autor

Nota: Nunca guardar un archivo como *picamera.py* ya que este nombre es usado por la librería.

Programa para crear rutina de captura de imágenes, a fin de poder realizar la captura de imágenes de manera continua y fluida; se crea un programa en python, en el cual mediante la implementación de un lazo “**for**” se puede cambiar a voluntad o conveniencia la cantidad de: capturas consecutivas a realizar, el intervalo de tiempo entre ellas (a fin de poder modificar los patrones para una misma clase de contaminante entre una muestra y otra) y la ubicación de las imágenes tomadas. En la figura 3.7 se muestra el código para realizar 30 muestreos con una separación de 5 segundos entre uno y otro.

En la estructura del programa se nota primero la importación desde las librerías de las funciones **PiCamera** (necesaria para trabajar la cámara) y **sleep** que suspende o detiene un programa durante un número de determinado de segundos.

```
#librerias
from picamera import PiCamera
from time import sleep

#definimos camara
camera = PiCamera()
camera.rotation = 0

camera.start_preview(alpha=255) #150 es semitranslucido, 255 es solido
#range(fotos a tomar)
for i in range(30):
    sleep(5)
    camera.capture('/home/pi/Desktop/Solidos/image%s.jpg'%i)
    print("Se ha tomado la foto No." + str(i))
camera.stop_preview()
```

Figura 3.7: Código usado para facilitar la toma de muestras

Fuente: el autor

Nota: Más adelante se darán más detalles de las demás funciones usadas, ya que estas también fueron utilizadas en el programa principal del clasificador.

Obtención del Modelo

Uso de Lobe.- Una vez ya obtenidas las muestras; podemos generar el modelo de aprendizaje automático siguiendo estos pasos:

- **Crear un nuevo proyecto:** Ya descargada la aplicación, se abre dando doble clic sobre el icono creado en el escritorio y presionando la opción **New project** en el entorno principal, el cual se muestra en la figura 3.8, automáticamente se abrirá una nueva ventana donde se podrá nombrar al proyecto y comenzar a trabajar con él; las opciones y característica de esta pantalla se muestra en la figura 3.9.

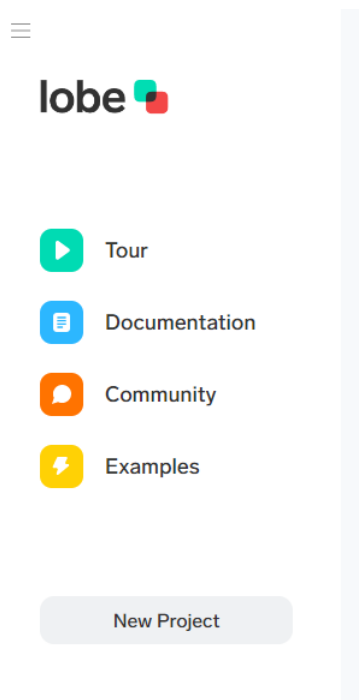


Figura 3.8: Opciones pantalla principal de Lobe

Fuente: el autor

Una vez ya creado y nombrado el nuevo proyecto como Clasificador de contaminantes (Pollutans Classifier); se procedió a cargar las muestras recogidas, etiquetándolas.

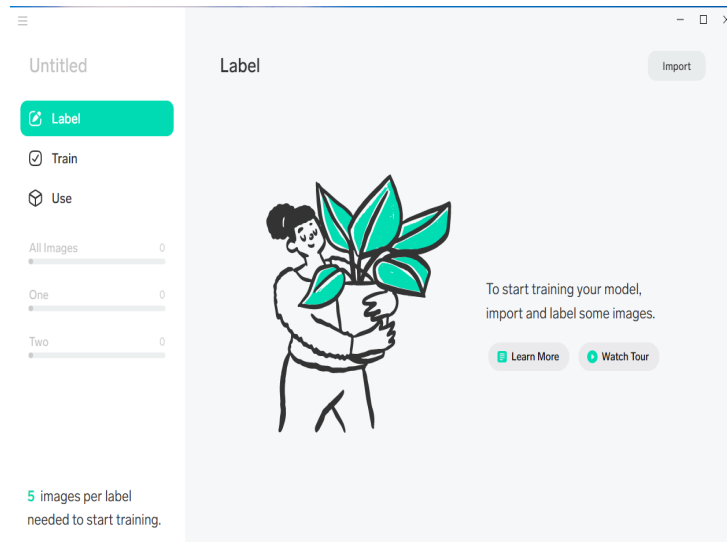


Figura 3.9: Pantalla Nuevo proyecto de Lobe

Fuente: el autor

- **Etiquetar:** Consiste en asignar categorías a las imágenes de las muestras tomadas, para crear ejemplos que enseñen a Lobe (estos ejemplos se conocen comúnmente como conjunto de datos). Se debe etiquetar para cada predicción deseada; por eso se creó, una etiqueta para cada categoría que se desea que clasifique el modelo; utilizando la opción *Label* mostrada en 3.9.

A partir de este conjunto de datos, Lobe aprenderá a identificar automáticamente una etiqueta para una imagen determinada. En otras palabras se etiqueta las imágenes para crear un conjunto de datos que le enseñe a Lobe como resolver el problema. En este punto cabe anotar que Lobe tiene un requisito mínimo de 5 imágenes por etiqueta (categoría a predecir o clasificar).

Se puede corregir los errores de etiquetado directamente en los resultados; es decir, al encontrar algunas predicciones marcadas como incorrectas porque fueron ejemplos etiquetados erróneamente, se las puede cambiar haciendo clic en el cuadro de texto de la etiqueta identificada y escribiendo o seleccionando la etiqueta verdadera.

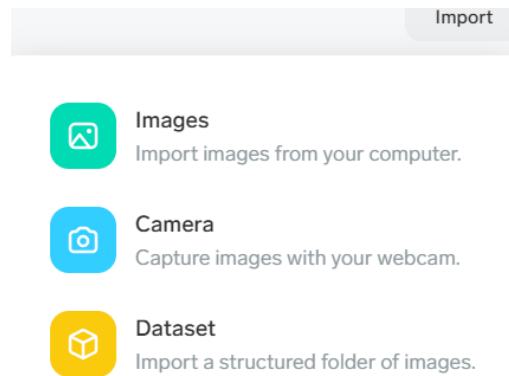


Figura 3.10: Opciones para importar imágenes

Fuente: el autor

- **Creación de Base de datos.-** Lobe maneja varias opciones para importar y etiquetar las imágenes, las cuales se pueden ver en el figura 3.10 y se detallan a continuación:

Images: Importa archivos de imágenes comunes directamente desde la computadora. Lobe admite los formatos JPEG, PNG, BMP y WebP.

Camera: Permite usar cualquier fuente de cámara conectada a nuestro ordenador para capturar imágenes directamente en Lobe (ej. cámara WEB). Opcionalmente, se puede proporcionar una etiqueta para estas imágenes previo a su captura (usando la opción Label); y si se mantiene presionado el botón de la cámara, se realiza la captura de una ráfaga de imágenes las cuales pertenecerán a esta misma categoría (tendrán igual etiqueta). A medida que una nueva imagen es capturada, el software emite el sonido de un clic y a la par el número de la muestras en la categoría va aumentando; finalmente al terminar de importar las imágenes tomadas se emitirá el sonido de una campana.

Dataset: Importa un conjunto de datos existente mediante el uso carpetas previamente creadas con las imágenes que pertenecen a una clase, importa las imágenes etiquetadas existentes utilizando los nombres de las carpetas como etiquetas. Puede crear nuevas etiquetas o editar las existentes usando el cuadro de texto marcado como "Label" en la esquina inferior de cada imagen como se muestra en la figura 3.11.

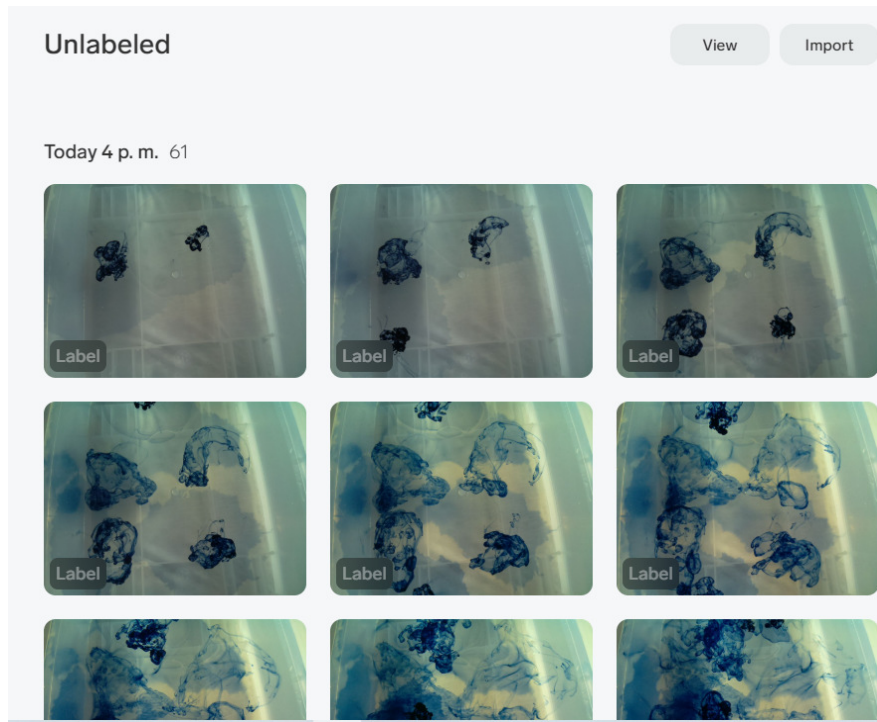


Figura 3.11: Muestras agua turbia sin etiquetar ya Importadas

Fuente: el autor

Nota: El tamaño máximo de imagen que puede procesar Lobe es 178'956.970 píxeles. Para una imagen cuadrada, eso es aproximadamente 13.300 x 13.300 píxeles.

- **Entrenamiento:** En esta etapa se le enseña al modelo de aprendizaje automático a predecir las etiquetas correctas a partir de sus ejemplos. Se puede pensar en los ejemplos como una colección de tarjetas didácticas; durante el entrenamiento, el modelo revisará continuamente las tarjetas e intentará encontrar patrones similares que le ayuden a identificar o clasificar las respuestas correctas. Lobe comienza a entrenar automáticamente cuando sus ejemplos cumplen con los requisitos mínimos. Para comenzar a entrenar, necesita:
 - Imágenes importadas para etiquetar como ejemplos
 - Al menos dos etiquetas
 - Al menos cinco imágenes por etiqueta

Una vez cumplido estos requisitos aparecerán al mismo tiempo el mensaje Lobe esta empezando a entrenar en la parte inferior, el signo de un visto que esta apareciendo y desapareciendo constantemente y un icono circular al lado de la opción entrenamiento, que se irá llenando a medida que avanza el proceso; una vez cumplido cierto porcentaje del entrenamiento, el mensaje cambia mostrando ahora el porcentaje de predicciones correctas e incorrectas como los muestra la figura 3.12. Una vez finalizado el entrenamiento se completará el círculo y se escuchará el sonido de una campana.

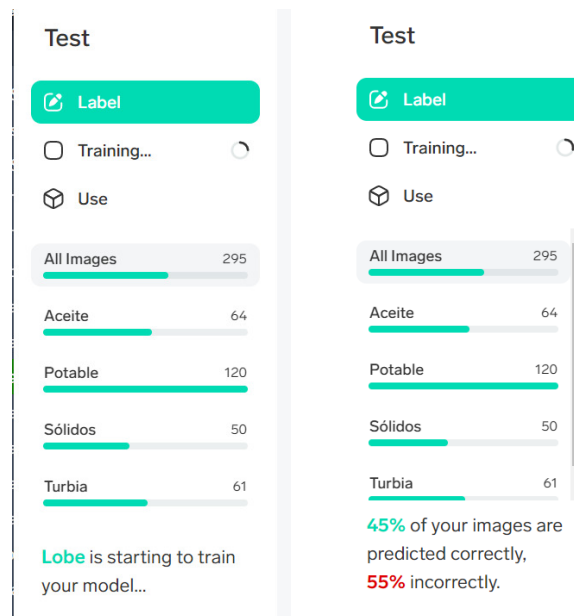


Figura 3.12: La Etapa de entrenamiento muestra sus avances en porcentaje

Fuente: el autor

Nota: Finalizado el entrenamiento automático, se puede optimizar manualmente el modelo para entrenar durante más tiempo y obtener un mejor rendimiento en el mundo real, presionando la opción **Optimize Model...** siguiendo la ruta *Archivo > Optimizar modelo*.

Mientras el modelo se está entrenando Lobe sigue las mejores prácticas para continuar el entrenamiento cuando se realice cambios en los ejemplos. Al realizar estos cambios o al agregar / eliminar etiquetas, Lobe restablecerá el entrenamiento y comenzará a entrenar un nuevo modelo de manera automática.

Cabe anotar que el tiempo de entrenamiento es bastante variable y depende de lo difícil que sea distinguir entre las etiquetas en los ejemplos (muestras o datos) y cuántos ejemplos se tiene. Puede llevar desde minutos hasta horas y a veces, días para problemas muy grandes.

Además, se puede colocar el cursor sobre el progreso del entrenamiento para ver una estimación del tiempo que este podría tomar. Esta estimación del tiempo de entrenamiento se actualiza cada pocos segundos en función de su progreso y la velocidad de procesamiento actual de la computadora. Es posible que se vea que fluctúa si se está realizando otras tareas en la computadora a medida que la CPU y la memoria disponibles cambian.

Lobe elige automáticamente la mejor arquitectura para encontrar el modelo automático en nuestro problema; no se necesita instalación ni configuración, la opción de configuración del proyecto (**Configure**) que se encuentra siguiendo la ruta *Archivo > Configuración del proyecto*; permite cambiar el proyecto entre los dos modos siguientes:

- **Precisión:** Un modelo más grande que generalmente logrará una mayor precisión en problemas más difíciles, pero tendrá tiempos de predicción más largos y utilizará más memoria.
- **Velocidad:** Un modelo más pequeño que tendrá una velocidad de predicción más rápida y un menor uso de memoria, pero puede tener menor precisión. Este modelo también se puede optimizar para dispositivos periféricos como teléfonos móviles o Raspberry Pi.

El cambiar el modo de configuración del proyecto reiniciará cualquier entrenamiento completado hasta el momento y entrenará automáticamente un nuevo modelo. La ruta y opciones de configuración aparecen en pantalla tal como se observa en la figura [3.13](#).

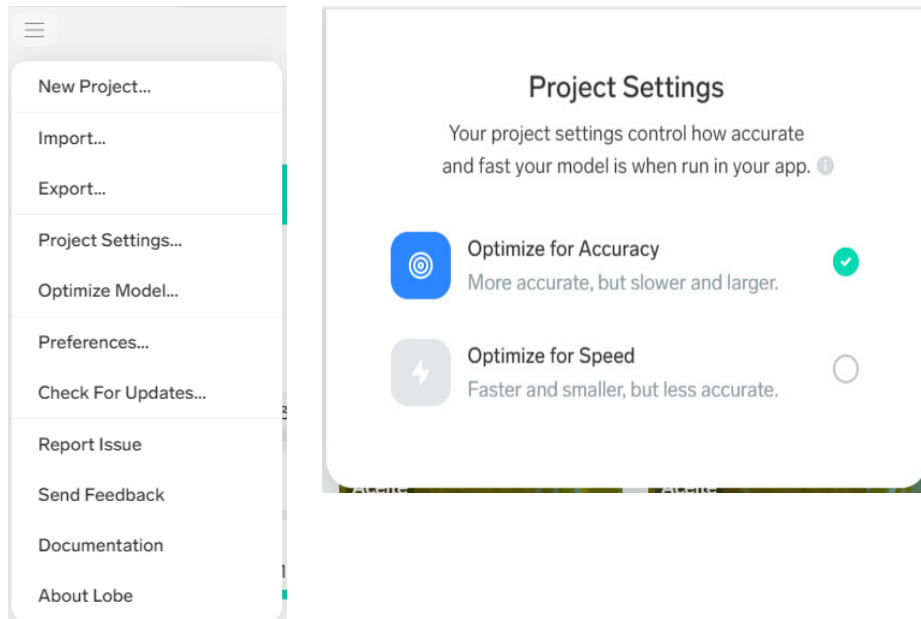


Figura 3.13: Opciones de Configuración del Proyecto

Fuente: el autor

Al cargar imágenes para entrenamiento, Lobe recortará el cuadrado central y escalará la captura resultante, además creará automáticamente pequeñas variaciones de las imágenes para reflejar el ruido de los datos en el mundo real. Durante el entrenamiento, Lobe hará cinco variaciones en las imágenes con cambios realizados aleatoriamente en:

- **Brillo**
- **Contraste**
- **Saturación**
- **Matiz**
- **Rotación**
- **Zoom**
- **Saturación**
- **Ruido de codificación JPEG**

- **Obtención de Resultados**, una vez ya realizado el entrenamiento se evaluó las fortalezas y debilidades del modelo obtenido para encontrar áreas de mejora.

Al presionar “**All Images**” aparecen todas las imágenes con la evaluación de su predicción, los resultados muestran que imágenes predice el modelo de forma correcta e incorrecta. Las predicciones correctas tienen etiquetas verdes y las predicciones incorrectas tienen etiquetas rojas, tal como se muestra en la figura 3.14.

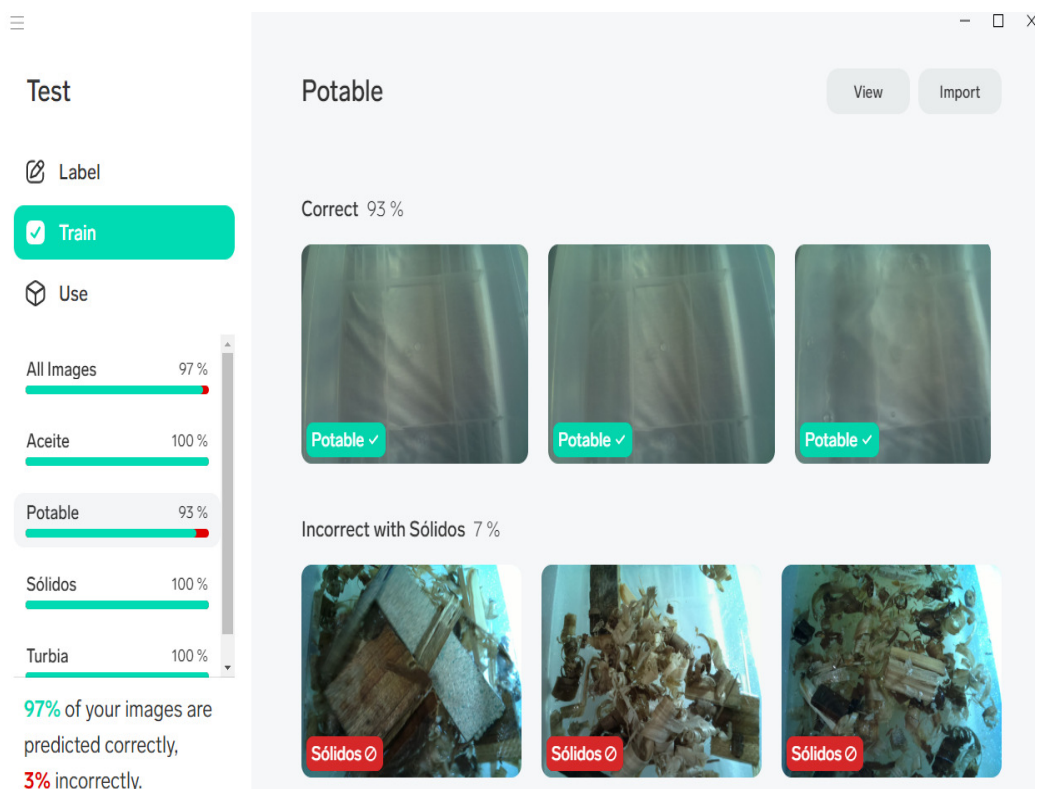


Figura 3.14: Resultados de predicciones

Fuente: el autor

El ancho de la barra de etiquetas representa la confianza que tenía el modelo en esa predicción, al pasar el cursor sobre una etiqueta predicha, se mostrará la verdadera etiqueta que se le dió a esa imagen. Cuanto más aciertos haya en las predicciones, mejor será el rendimiento del modelo.

Se puede ver y ordenar las imágenes de diferentes formas utilizando la opción “**View**” o la barra lateral para comprobar:

- Si el modelo está aprendiendo correctamente todas las etiquetas con *View > All Images*.
- Aproximadamente que tan bien funcionará con nuevas imágenes seleccionadas con *View > Test Images*.
- Que imágenes y etiquetas confunden a su modelo al seleccionar las etiquetas individuales en la barra lateral.

En la figura 3.15 se muestra la ventana emergente de la opción “**View**”.

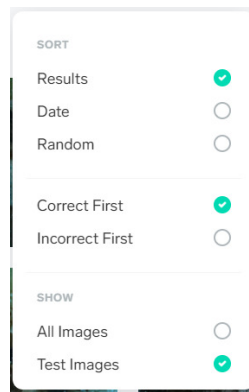


Figura 3.15: Opciones de visualización de resultados de la predicción

Fuente: el autor

Como pautas para ver donde está más confundido el modelo, se puede observar las predicciones incorrectas con mayor probabilidad y las predicciones correctas con menos probabilidad como sigue:

- Seleccionar *View > Correct first* para ver las predicciones ordenadas desde la más confiable a la menos confiable. Aquí se intenta encontrar cualquier patrón en el que el modelo base su predicción incorrecta.
- Seleccionar *View > Incorrect first* para ver las predicciones ordenadas desde la menos segura a la más segura. Aquí se ve donde el modelo es menos confiable con sus predicciones correctas.

En este punto se recopila imágenes con más variaciones que tengan patrones similares a estos dos casos para ayudar a mejorar su modelo.

Test Images, las imágenes de prueba son un subconjunto aleatorio de los ejemplos que Lobe oculta del modelo durante el entrenamiento. Lobe divide automáticamente sus ejemplos en dos partes:

Se utiliza un 80 % aleatorio para entrenar el modelo y el 20 % aleatorio se reserva y usa para probar el modelo. Se puede ver los resultados obtenidos solo de las imágenes de prueba al seleccionar: **View > Test Images**. Si el rendimiento general en todas las imágenes es muy alto y con las imágenes de prueba es menor, es posible que el modelo esté memorizando los ejemplos (datos de entrenamiento), en lugar de aprender a generalizar a imágenes invisibles. A esto se le suele llamar sobreajuste. Para ayudar a prevenir el sobreajuste, se puede:

- Recopilar más imágenes en general para que el conjunto de entrenamiento incluya más variaciones.
- Asegúrese de que las imágenes no se parezcan demasiado entre sí.

Si se selecciona las etiquetas individuales en la barra lateral para ver los resultados específicos de la etiqueta; Lobe mostrará que otras etiquetas se confunden comúnmente con esta, como se muestra en la figura 3.16.

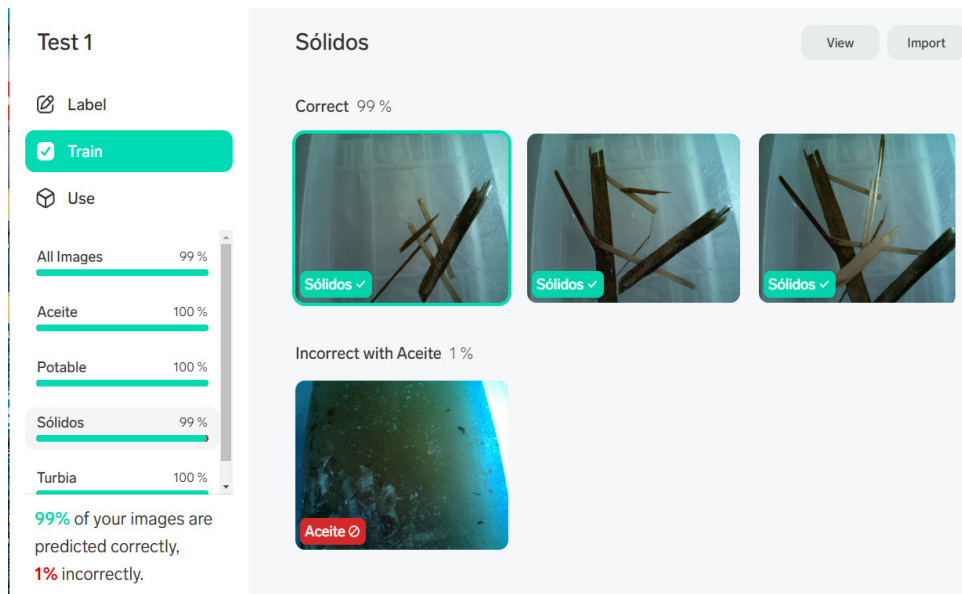


Figura 3.16: Ejemplo al seleccionar etiqueta individual **Sólidos**.

Fuente: el autor

En este punto se intenta detectar que similitudes existen entre las imágenes que confunden al modelo y las imágenes en la etiqueta verdadera. Por ejemplo, se podría notar el mismo color de fondo en las imágenes confusas y en las imágenes de la etiqueta seleccionada, o que no se alcanza a distinguir con claridad la característica deseada. Una vez ya identificados estos patrones, se recopiló diferentes imágenes que se parezcan a la imagen confusa para enseñarle mejor a Lobe que patrones ignorar. En el ejemplo de la figura 3.16, se necesita recopilar más imágenes en la etiqueta seleccionada con sólidos que cubran el centro de la captura en poca área. Además se eliminó el aserrín como elemento para simular sólidos, ya que además de ser poco visible, este tenía el efecto negativo de ennegrecer el agua de la muestra.

Ya satisfecho con los resultados, se puede probar el modelo; para esto se usó nuevas imágenes seleccionando la opción **Use**; pudiendo escoger entre las categorías: *Images* (permite escoger una imagen desde el ordenador para la prueba), *Camera* (activa la cámara del ordenador para realizar una captura) o *Export* (exporta el modelo para usarlo en la aplicación deseada). En la figura 3.17 se observa una prueba del modelo utilizando la opción “*Use*” con “*Camera*” la en camera.

Con el fin de probar el modelo se lo intentó engañar con nuevas imágenes que detecten sus límites o falencias y se agregó esas imágenes como ejemplos para mejorar el modelo, en la figura 3.17 se muestra una prueba del modelo hecha con una imagen que no pertenece a la base de datos usada en el entrenamiento, para este caso el modelo acertó la predicción mostrando “Aceite” como respuesta a la prueba, para agregarla a la base de datos como muestra de aceite se da clic en el recuadro verde que tiene un visto, en caso de que la predicción hubiera sido incorrecta se da clic en el recuadro rojo y aparece un submenu con todas las etiquetas, se escoge la correcta para después añadir la imagen a nuestra base de datos.

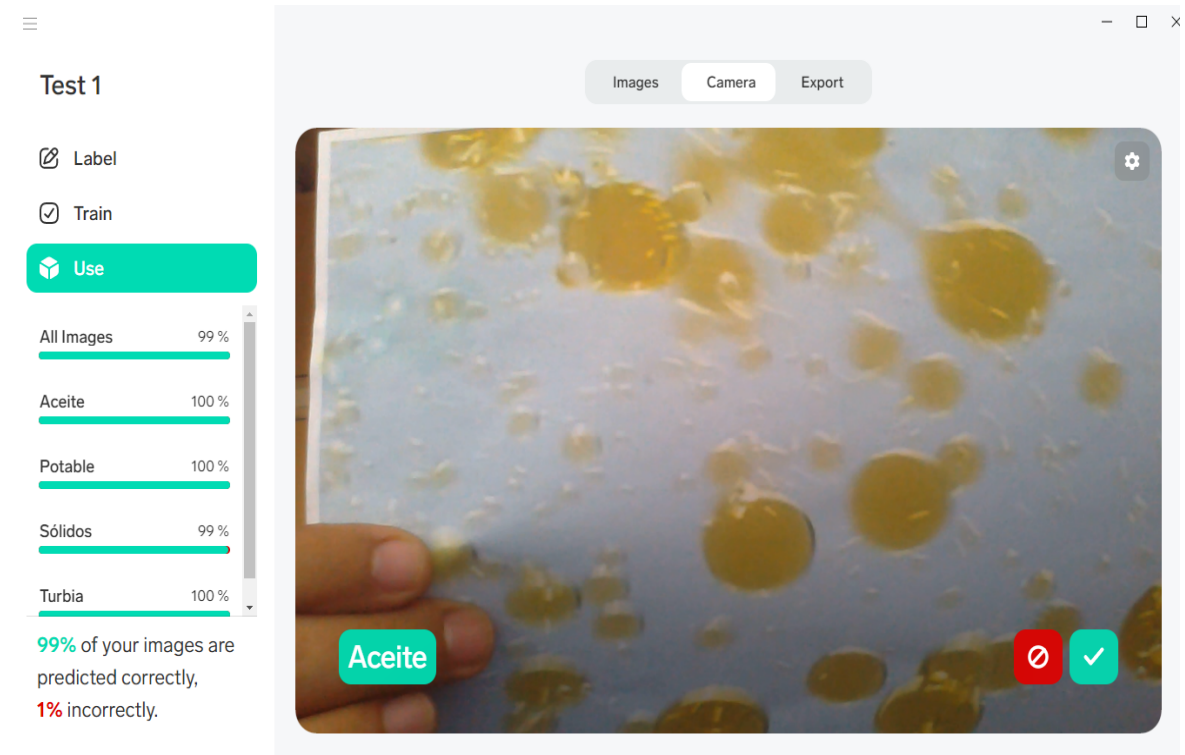


Figura 3.17: Prueba del modelo con con Imagen fuera de la muestra **Aceite**.

Fuente: el autor

- **Exportar**, Después de que los resultados obtenidos con el modelo son satisfactorios, se lo puede exportar en formatos estándares para la industria; con el fin de usarlo en nuestra aplicación. Entre ellos tenemos:
 - TensorFlow, para usarlo en aplicaciones codificadas en lenguaje Python.
 - TensorFlow Lite, para usar el modelo en aplicaciones de Android e Internet de las cosas.
 - Core ML, para desarrollar aplicaciones con iOS, iPad y macOS

Básicamente, el modelo es una colección de archivos que otros programas pueden cargar para ejecutar predicciones. Estos archivos almacenan tanto la estructura del modelo como los pesos que son el resultado del entrenamiento.

Se Puede utilizar los archivos del modelo localmente en nuestra aplicación o en la mayoría de las principales plataformas en la nube para crear una API. Lobe también aloja el modelo como una API local para ayudar a impulsar el desarrollo de la aplicación.

La herramienta Lobe, proporciona algunos flujos de trabajo para usar los modelos obtenidos; como llamar a una API local, adaptar proyectos de inicio o trabajar con archivos de modelo directamente. En otras palabras Lobe nos permite:

Integrar con una aplicación externa, si se está utilizando Lobe con otra aplicación, se necesita comprobar si se puede llamar a una API; para obtener predicciones a través de una red o si puede cargar y ejecutar archivos del modelo directamente.

- **Usar una API para predicciones:** Si la aplicación deseada puede realizar solicitudes POST, procesar JSON y codificar imágenes en base64, puede crear una solicitud de red para obtener predicciones de su modelo.
- **Cargar archivos de modelo directamente:**
Si la aplicación deseada puede cargar y ejecutar modelos externos, se debe seleccionar la opción de exportación de archivo de modelo adecuada para la aplicación.

Crear una API para mayor flexibilidad, producir una API REST es una de las formas más versátiles de conectar aplicaciones para usar un modelo. Se puede llamar al modelo como un servicio desde otros productos, implementarlo en una variedad de proveedores de nube o incluso ejecutarlo en nuestra propia computadora o dispositivo periférico como una Raspberry Pi.

Formas en que se puede utilizar nuestro modelo: Entre otras podemos mencionar.

- **Aplicación de Python local o alojada en Azure, Google Cloud o AWS**, se exporta el modelo como TensorFlow. El modelo guardado de TensorFlow es un formato estándar que se usa en las aplicaciones de Python que ejecutan TensorFlow 1.x o 2.x, y se puede implementar en los servicios web de TensorFlow para ejecutar inferencias en la nube como una API.
- **Android o Raspberry Pi**, se exporta el modelo como TensorFlow Lite para usarlo en aplicaciones móviles y de IoT. Se debe tener en cuenta de utilizar el modelo de optimizado por velocidad desde *Archivo > Configuración del proyecto*; si se necesita baja latencia y un menor uso de memoria.

Para nuestro caso usamos el formato TensorFlow Lite, para exportar nuestro modelo; una vez que se le ha asignado el nombre y la dirección, Lobe crea una carpeta con el modelo exportado, que a la vez contiene una subcarpeta con un código de ejemplo para usar el modelo (*example*); además de 3 archivos (*labels.txt*; *saved_model.tflite* y *signature.json*).

example/ <carpeta>: En esta, Lobe proporciona un ejemplo con un archivo *readme*; sobre como cargar y ejecutar el modelo en el formato de exportación elegido.

signature.json <archivo>: La firma contiene información sobre el modelo que la aplicación generalmente deberá proporcionar cuando lo cargue. Esto incluye detalles sobre las entradas y salidas expuestas, los tipos y formas de datos, y las etiquetas correspondientes a los resultados de confianza (probabilidad) predichos. Se puede consultar el directorio */ example* para ver el código de como cargar y usar la firma.

labels.txt <archivo>: En este fichero se encontrará las etiquetas definidas durante el entrenamiento.

Diseño e Implementación del Clasificador

Descripción.- Para la implementación del clasificador se decidió crear un prototipo; que por medio de un circuito eléctrico y a través de una salida digital digital (LED), defina a que grupo de los contaminantes corresponde la imagen tomada de la muestra, en la figura 3.18 se muestra el diagrama de bloques con los diferentes componentes.

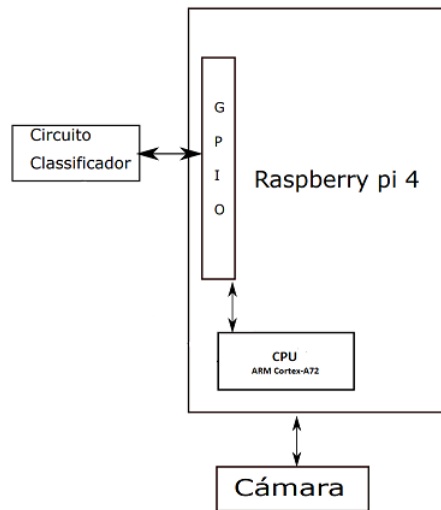


Figura 3.18: Diagrama de Bloque de la Implementación.

Fuente: el autor

Como ya se mencionó; se decidió usar la Raspberry Pi 4 con su módulo de cámara para capturar, guardar y procesar (clasificar), la imagen de la muestra a evaluar; por esto se procedió a asignar una salida digital por cada etiqueta (clase de contaminante) y dos más para señales de status, asignando a cada una un led de distinto color a fin de que el resultado pueda ser identificado con facilidad, quedando su distribución como sigue:

Para las Clases:

Aceite ->Amarillo; Potable ->Azul; Turbia ->Rojo; Naranja ->Sólidos.

Para Estatus:

Verde ->ON / OFF; Blanco ->Captura de Foto.

Procedimiento.-Una vez ya obtenido y exportado el modelo seguimos los siguientes pasos:

Descarga del Modelo: El contenido de la carpeta resultante de la exportación (modelo en formato Tensor Flow Lite) se copia en la MicroSD, dando la siguiente dirección `/home/pi/lobe/model`; a fin de ser desplegado desde la Raspberry y pueda ser usado por el programa que se encargará de ejecutar la rutina de comparación; y dar un resultado en la clasificación de la imagen capturada.

Previo a esto la Raspberry Pi fue actualizada mediante la ventana terminal usando el comando:

```
pi@raspberrypi:~ $ sudo apt-get update
```

y también se procedió a la instalación de la biblioteca TensorFlow Lite runtime usando la orden:

```
pi@raspberrypi:~ $ pip3 install tensorflow
```

Luego, se instaló la librería Lobe-python para Python3 ejecutando el siguiente comando de bash:

```
pi@raspberrypi:~ $ sudo bash lobe-rpi-install.sh
```

Programación para rutina de clasificación

- A fin de habilitar las GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General); como botón pulsador e indicadores leds en la Raspberry; se importan los métodos Button, LED, PWMLED, PiCamera y sleep desde las librerías respectivas; así también la función ImageModel, como se muestra en la figura 3.19.

```
#import Pi GPIO library button class
from gpiozero import Button, LED, PWMLED
from picamera import PiCamera
from time import sleep

from lobe import ImageModel
```

Figura 3.19: Carga de librerías y métodos.

Fuente: el autor

- Se asigna salidas digitales para cada una de las categorías de contaminantes creadas, a entender: Aceite, Potable, Sólidos y Turbia, así como dos salidas tipo PWMLED, una para indicar el estado del dispositivo y la otra para indicar el estado en el proceso de captura de imagen. Además se crea (habilita) la camera y se asigna una GPIO para el pulsador. Como se muestra en la figura 3.20.

```
#Create input, output, and camera objects
button = Button(2)

yellow_led = LED(17) #Aceite
blue_led = LED(27) #Potable
orange_led = LED(22) #Sólidos
red_led = LED(23) #Turbia
white_led = PWMLED(24) #Status light and retake photo
power_led = PWMLED(25) #Indicador ON/OFF(green)
camera = PiCamera()
```

Figura 3.20: Creación y asignación de entradas y salidas.

Fuente: el autor

- Se carga el modelo desde la dirección previamente establecida, este paso se detalla en la figura 3.21

```
# Load Lobe TF model
# --> Change model file path as needed
model = ImageModel.load('/home/pi/Lobe/modeloCont')
```

Figura 3.21: Carga del Modelo.

Fuente: el autor

- Se declara la función `take_Photo`, como se muestra en la figura 3.22 aquí se definen:
 - El nivel de opacidad con que se muestra la vista previa, usando el parámetro *alpha* el cual puede ir de 0 a 255.
 - El tiempo que se toma desde que inicia la vista previa hasta que se captura la imagen empleando la función *sleep*.

Nota: Se considera fundamental pausar un tiempo mínimo de dos segundos previo a la captura de una imagen; de esta forma, se provee el tiempo suficiente al detector de la cámara para que pueda definir correctamente los índices de luminosidad.
 - En caso de requerir girar o rotar la imagen se puede usar el la opción *camera.rotation*.
 - Se captura la imagen en formato jpg y se la direcciona, se desactiva la imagen previa y el led indicador de captura de imagen se apaga.

```
# Take Photo
def take_photo():

    # Start the camera preview
    camera.start_preview(alpha=255)
    # wait 2s or more for light adjustment
    sleep(3)
    # Optional image rotation for camera
    # --> Change or comment out as needed
    camera.rotation = 0
    #Input image file path here
    # --> Change image path as needed
    camera.capture('/home/pi/Pictures/image.jpg')
    #Stop camera
    camera.stop_preview()
    white_led.off()
```

Figura 3.22: Definición de la función `take_photo`.

Fuente: el autor

- Se define la rutina de identificación de la predicción y encendido apropiado de los leds, como se muestra en la figura 3.23.

```
# Identify prediction and turn on appropriate LED
def led_select(label):
    print(label)
    if label == "Aceite":
        yellow_led.on()
        sleep(5)
    if label == "Con Sólidos":
        orange_led.on()
        sleep(5)
    if label == "Potable":
        blue_led.on()
        sleep(5)
    if label == "Turbia":
        red_led.on()
        sleep(5)
        red_led.off()
    else:
        yellow_led.off()
        blue_led.off()
        orange_led.off()
        red_led.off()
        white_led.off()
```

Figura 3.23: Definición de la función led_select.

Fuente: el autor

- Se declara la función principal con su secuencia de ejecución como sigue:

El led de status On/Off (verde) enciende y queda energizado. Dentro de un bucle *While True*, se crea la rutina para la captura de la imagen; de tal manera que al presionar el pulsador (GPIO 2), el led de estado de la cámara (blanco) comienza a titilar con una frecuencia de 0,5 s, se ejecuta la función *take_foto* (descrita anteriormente), se pausa medio segundo y el led blanco comienza a parpadear más rápido con frecuencia de 0,1 s indicando que inicia el proceso de comparación y clasificación de la foto tomada; después el led blanco se apaga y se enciende el led de predicción de acuerdo al resultado de la clasificación obtenida, finalmente hay una pausa de 1 s para volver a iniciar la secuencia, la figura 3.24 muestra la programación de la función principal.

```

# Main Function
power_led.on()
while True:
    if button.is_pressed:
        white_led.pulse(0.5,0.5)
        take_photo()
        # Run photo through Lobe TF model
        sleep(0.5)
        white_led.pulse(0.1,0.1)
        result = model.predict_from_file('/home/pi/Pictures/image.jpg')
        white_led.off()
        led_select(result.prediction)
    sleep(1)

```

Figura 3.24: Definición de la función principal.

Fuente: el autor

Nota: El programa usado se muestra de manera integral en el apéndice A

Circuito Eléctrico

Antes de diseñar el circuito, debemos conocer como se configura la entradas y salidas de la Raspberry Pi (esto ya se realizó con la programación). En la figura 3.25 se puede ver la asignación de pines para una Raspberry Pi 4.

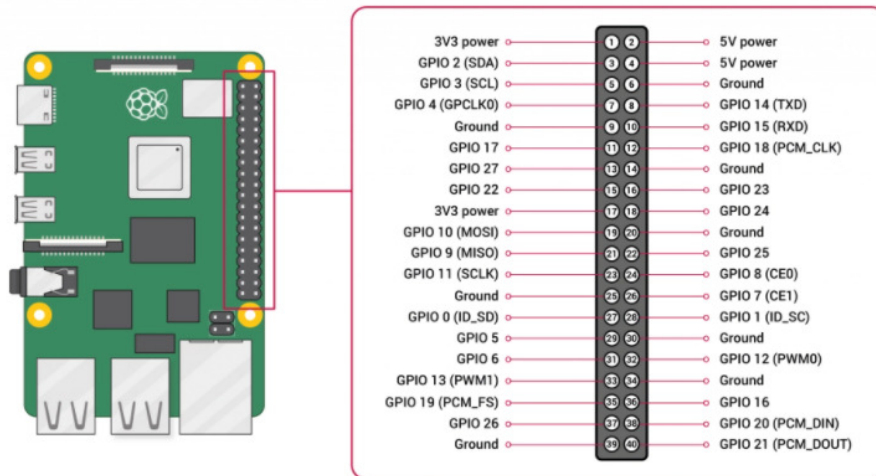


Figura 3.25: Esquema de Pines GPIO de la Raspberry

Fuente: [Isaac, 2020]

A continuación se detalla la forma en que se conectó el botón pulsador y los diodos LEDS a los pines GPIO de la Raspberry Pi 4:

- Botón pulsador: Se conecta un terminal del botón pulsador al pin de la GPIO 2 y el otro mediante una resistencia, a un pin GPIO de GND.
- LED amarillo: El terminal positivo (más largo) al pin de la GPIO 17 y el otro a través de una resistencia, a un pin GPIO de GND.
- LED azul: El terminal positivo al pin de GPIO 27 y el otro a través de una resistencia, a un pin GPIO de GND.
- LED rojo: El terminal positivo al pin de la GPIO 23 y el otro a través de una resistencia, a un pin GPIO de GND.
- LED naranja: El terminal positivo al pin de la GPIO 22 el otro a través de una resistencia, a un pi GPIO de GND.
- LED verde: El terminal positivo al pin de la GPIO 25 y la otro, a través de una resistencia, a un pin GPIO de GND.
- LED blanco: El terminal positivo al pin de la GPIO 24 y el otro a través de una resistencia, a un pin GPIO de GND.

La forma del cableado y las conexiones se muestran en la figura 3.26.

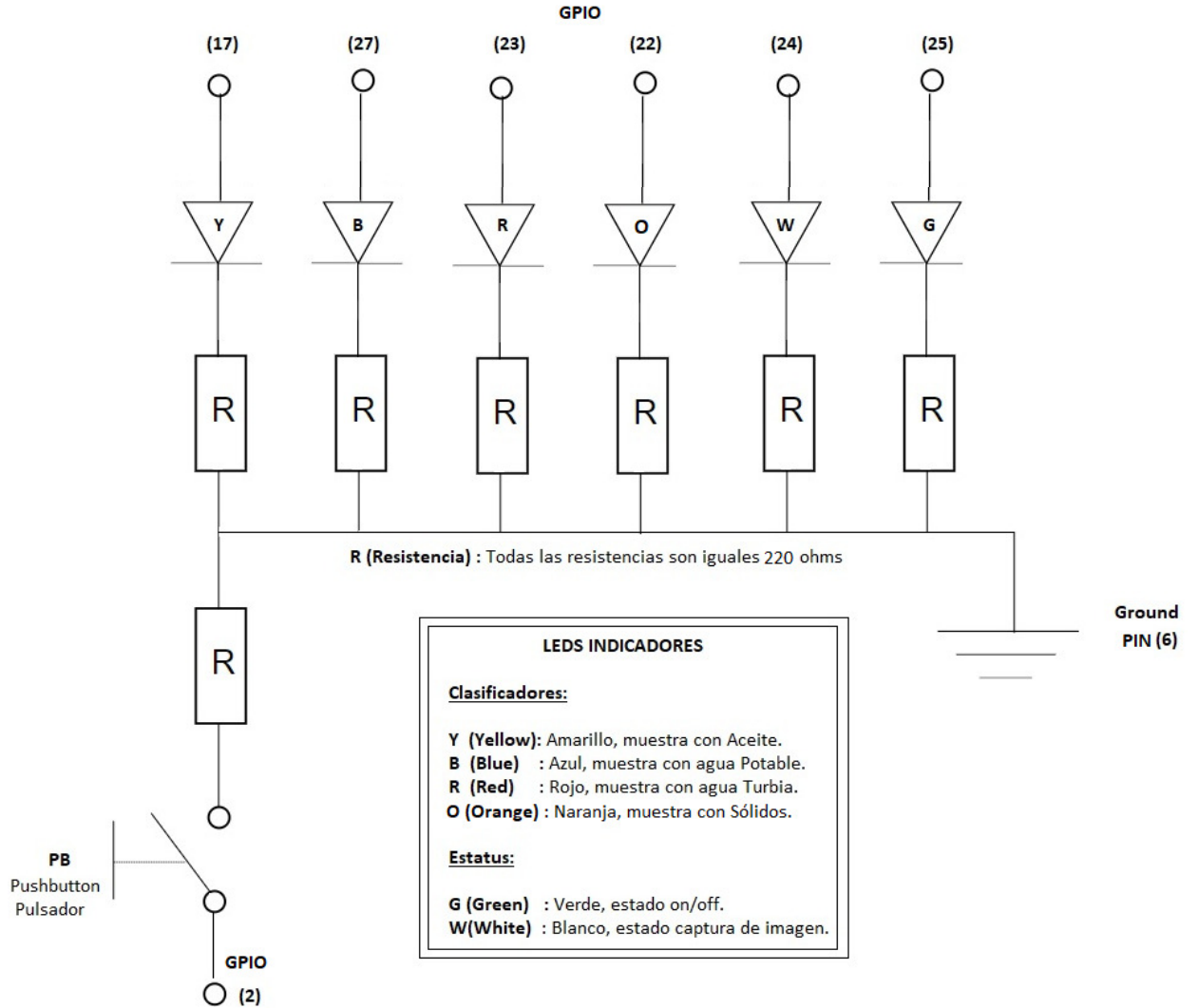


Figura 3.26: Esquema de conexiones del circuito, las GPIOs y Pin usados están entre paréntesis

Fuente: el autor

Se utilizó un protoboard pequeño para desarrollar el circuito en la etapa de prueba.

Se hicieron todas las conexiones con la raspberry previamente cargada con el programa y el modelo; se la montó en la mesa de prueba, se ejecutó el programa y se verificó su correcto funcionamiento usando algunas muestras nuevas. Se muestra el prototipo siendo probado con una foto de sólidos en la figura 3.27 y con una muestra de agua turbia en la figura 3.28.

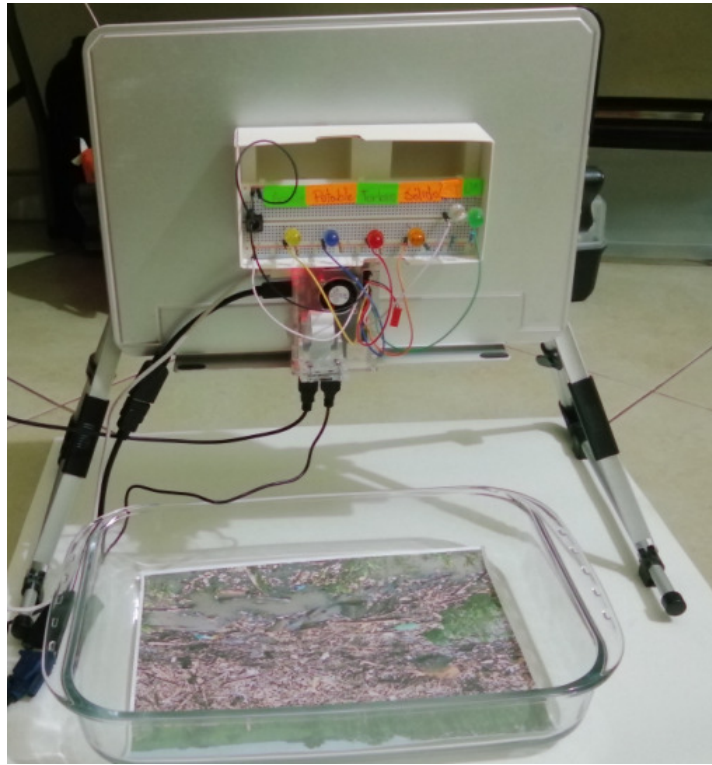


Figura 3.27: Prueba Prototipo modelo 3 con Foto.

Fuente: el autor

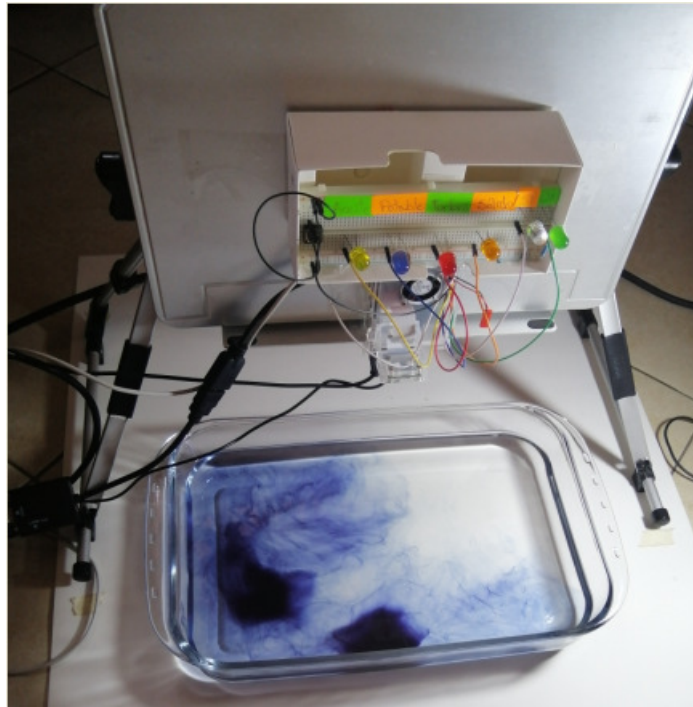


Figura 3.28: Prueba Prototipo modelo 3 con muestra Agua turbia.
Fuente: el autor

Capítulo 4

Resultados y Discusión

En el presente capítulo se presentan y analizan los resultados de las diferentes métricas para clasificadores multiclases, obtenidas al usar el modelo final generado por Lobe. Para esto se usará un conjunto de prueba consistente en imágenes que no pertenecen al conjunto de datos utilizados en el entrenamiento del modelo.

Las métricas evaluadas son las descritas en el capítulo 2, en definiciones previas, a entender:

- Matriz de confusión o error
- Exactitud
- Precisión
- Exhaustividad o sensibilidad
- Especificidad o TNR (Tasa Negativa Verdadera)
- F1-Score
- AUC

4.1. Evaluación Métricas utilizadas

A fin de calcular el rendimiento del modelo se obtiene la matriz de confusión, utilizando el conjunto de prueba (datos previamente etiquetados y no usados durante el entrenamiento); el cual como se mencionó anteriormente, Lobe lo escoge de manera aleatoria y representa el 20% del total del conjunto de datos, para nuestro caso consiste en 26 muestras por cada categoría; los resultados obtenidos se aprecian en la figura 4.1

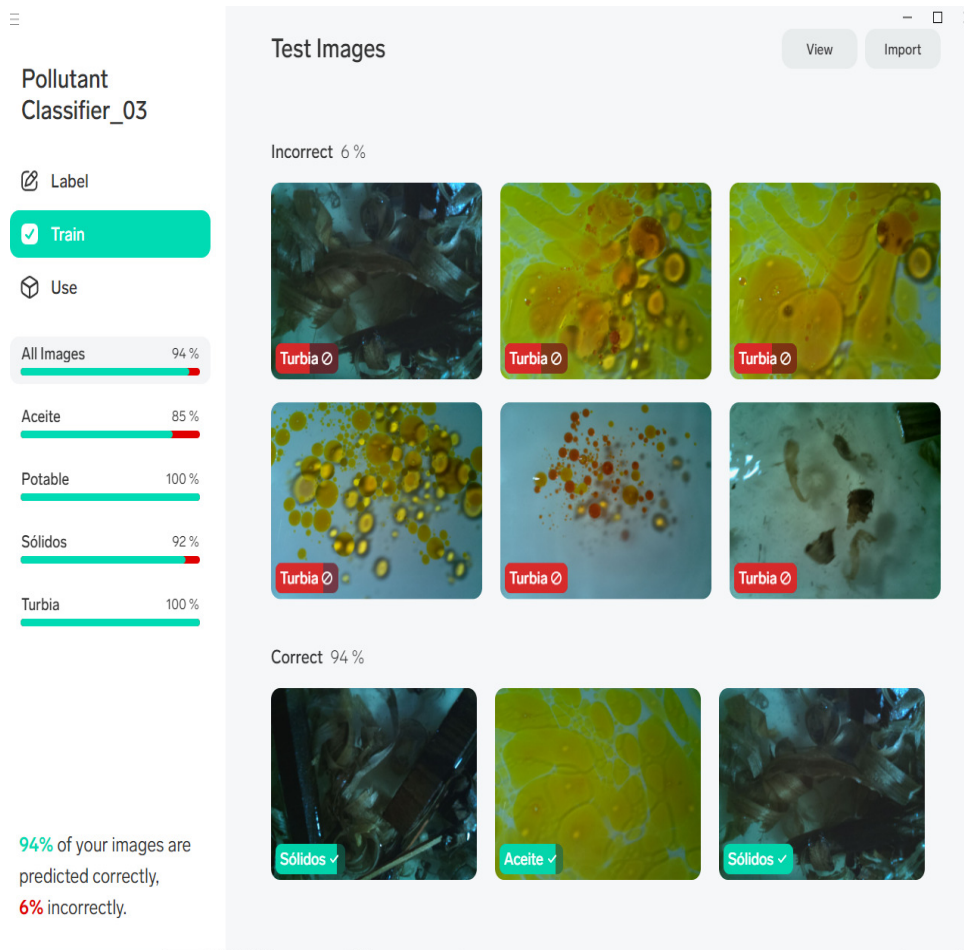


Figura 4.1: Imágenes de Test con error en su predicción.

Fuente: el autor

4.1.1. Matriz de confusión

Generamos la Matriz de confusión tomando los datos del test realizado por el software, los resultados se muestran en la tabla 4.1

Tabla 4.1: Matriz de Confusión.

		<i>PREDICCIÓN</i>			
		Clases	Aceite	Potable	Sólidos
<i>R</i>	Aceite	22	0	0	4
<i>E</i>	Potable	0	26	0	0
<i>A</i>	Sólidos	0	0	24	2
<i>L</i>	Turbia	0	0	0	26

Se puede observar que de las muestras Agua Potable y Turbia fueron clasificadas correctamente en su totalidad; mientras que en las muestras de Agua con Aceite, cuatro (4) fueron etiquetadas incorrectamente como Turbia y para las muestras de Sólidos, dos (2) fueron etiquetadas erróneamente como Turbia.

4.1.2. Tabla de Resultados de Métricas

Para este cálculo se usan las formulas expuestas en el capítulo 2 teniendo en cuenta lo siguiente para un clasificador multiclase (más de 2 categorías o etiquetas):

FN (Falsos Negativos).- Para una clase determinada es la suma de los valores de la fila correspondiente a está clase excluyendo el valor de la diagonal principal (VP: Verdadero positivo).

FP (Falsos Positivo).- Para una clase determinada viene dada por la suma de los valores de la columna correspondiente a está clase excluyendo el valor de la diagonal principal (VP: Verdadero positivo).

VN (Verdaderos Negativos).- Para cierta clase será la suma de todos los elementos de las columnas y filas excluyendo los elementos de fila y columna de la clase seleccionada.

Además, para obtener la curva ROC se necesita, el valor de la probabilidad de cada una de las capturas de imágenes pertenecientes al conjunto de evaluación con respecto a las otras clases; sin embargo Lobe, nos da valores de porcentaje diferentes de 100 % solo en las imágenes incorrectas respecto a la clase con la que se confunde y en 3 muestras correctamente categorizadas (dos muestras de sólido y una de aceite); esto se detalla en la tabla 4.2

Tabla 4.2: Probabilidades de muestras relevantes del Test (diferente de uno).

Muestra	Clase	Predicción	Probabilidad
1	Aceite	Turbia	1
2	Aceite	Turbia	0,76
3	Aceite	Turbia	0,6
4	Aceite	Turbia	0,58
5	Aceite	Aceite	0,89
6	Sólido	Turbia	1
7	Sólido	Turbia	0,54
8	Sólido	Sólido	0,99
9	Sólido	Sólido	0,88

Así obtenemos la tabla 4.3

Tabla 4.3: Resultado Clasificadores.

Clases	Precisión	Sensibilidad	Especificidad	F1
Aceite	1	0,8462	1	0,9167
Potable	1	1	1	1
Sólidos	1	0,9231	1	0,9600
Turbia	0,8125	1	0,9231	0,8966
<i>Exactitud</i>			<i>0,9423</i>	
<i>AUC</i>			<i>0,9988</i>	

Nota: Las formulas usadas para el calculo fueron descritas en el capítulo 2, estas son; (2.2) Exactitud; (2.3) Precisión; (2.4) Sensibilidad; (2.5) Especificidad; (2.6) F1 y (2.8) AUC.

4.2. Discusión de Resultados

Se observa que los errores de predicción en el conjunto de prueba fueron de 6 muestras de un total 104, lo que representa un porcentaje de error de 5,8 % aproximadamente; que indica una exactitud para el modelo de 94,2 %, que es alta; y ya que la cantidad de muestras están balanceadas (tanto en el entrenamiento como para el test) eliminando de esta manera la posibilidad de algún sesgo, se considera que el modelo de predicción funcionará muy bien.

Nota: Si disponemos de un “dataset” con desequilibrio, tiende a generarse un valor de precisión elevado en la clase “mayoritaria” y una baja sensibilidad en la clase “minoritaria”. Para evitar esto debemos recurrir al balanceo de clases.[Barrios, 2021]

Dado que la precisión tiene ver con la calidad de la predicción; es decir el porcentaje de muestras que sean clasificado como correspondiente a una clase y en realidad pertenecen a esta, encontramos que las muestras de Aceite, Potable y Sólidos han sido clasificadas correctamente por el modelo, mientras que este se confunde más con la clase Turbia asignando 6 muestras a esta categoría cuando pertenecen a otras dos clases (aceite y sólidos); de requerir corregir o mejorar este porcentaje, deberíamos escoger más imágenes de Aceite y Sólidos con los patrones que confunden al modelo, poniendo mayor énfasis en las muestras de aceite.

Con respecto a la sensibilidad o Recall, que nos da el porcentaje de muestras de una clase que hemos sido capaces de identificar, el menor valor esta en el aceite; ya que de las 26 muestras de esta clase en el conjunto de datos para la prueba; 4 fueron identificadas de manera errónea (como turbia), para mejorar este valor (de ser necesario); tendríamos que revisar las muestras incorrectamente identificadas y determinar cuales son los elementos dentro de esta que se asemejan al de la muestra turbia y darle un mayor peso (obtener más muestras de Aceite con ese tipo de patrones similares).

Tanto la exhaustividad y la tasa negativa verdadera (TNR), muestran la cualificación de un clasificador para poder distinguir los casos positivos de los negativos. La exhaustividad se representa como la porción de verdaderos positivos, en tanto que la TNR, indica la porción de verdaderos negativos,[Barrios, 2021] que para nuestro está altamente relacionada.

Siendo el mejor valor de F1 (sintetiza la precisión y la exhaustividad en un único valor) para la muestra de agua Potable, podemos concluir que esta tiene la mejor relación (compromiso o correspondencia) entre la precisión y sensibilidad, esto debido a que no existe falsos positivos, ni falsos negativos, en otras palabras la predicción es 100 %, exacta al clasificar las muestras como perteneciente o no a esta clase.

El valor elevado de AUC se debe a que 95 muestras de las 104 (total de capturas escogidas para la evaluación del modelo), fueron clasificadas acertadamente con un porcentaje de probabilidad del 100 %; mientras que de las 9 restantes, 3 fueron acertadamente clasificadas con porcentajes altos (99, 89 y 88 % respectivamente). Por lo cual, tan solo 6 muestras influyeron de manera negativa generando disminución en el valor ideal de este indicador (1).

De manera general podríamos concluir lo siguiente:

Alta precisión y alta exhaustividad: El modelo de Machine Learning escogido controla estupendamente a la clase.

Alta precisión y baja exhaustividad: El modelo de Machine Learning escogido no determina muy bien la clase, no obstante una vez que lo hace es sumamente confiable.

Baja precisión y alta exhaustividad: El modelo de Machine Learning escogido identifica bien la clase, aunque también incorpora datos de la otra clase.

Baja precisión y bajo Exahustividad: El modelo de Machine Learning escogido no logra clasificar la clase correctamente. [Barrios, 2021]

Los valores obtenidos en las métricas para la evaluación del modelo del clasificador desarrollado son bastante buenos; por lo cual, concluimos que este tiene la eficacia necesaria para su implementación.

Capítulo 5

Conclusiones y Recomendaciones

En el presente capítulo se detallan a modo de conclusiones, la obtención de los objetivos, las ventajas y desventajas encontradas a partir de la realización del proyecto presentado. Así como sugerencias en el ítem de recomendaciones.

Cumplimiento Objetivo General

Se desarrolló un sistema de reconocimiento e identificación utilizando modelos de aprendizaje automáticos, para determinar la presencia de diferentes agentes contaminantes en una muestra de agua.

Cumplimiento Objetivos Específicos

Se revisó y analizó el estado del arte sobre sistemas de identificación de contaminantes en un espejo de agua, encontrando los avances y desarrollos más recientes hechos con tecnologías IA sobre la calidad del agua; así se sentó las bases conceptuales y metodológicas para el desarrollo del presente proyecto.

Se entrenó un modelo de aprendizaje automático para clasificar el tipo de contaminante presente en una muestra de agua (capturada en una imagen) mediante el uso de Lobe; ya definidos los tipos de contaminantes a evaluar (Aceite, Potable, Turbia y con Sólidos), se creó una base de datos de 130 imágenes por cada contaminante, que fueron el insumo del software para la creación del modelo automático del clasificador .

Se probó el modelo obtenido mediante el uso de muestras aleatorias para obtener su porcentaje de efectividad y mejorarlo mediante reentrenamiento, incluyendo nuevas muestras con los patrones que confunden al modelo.

Finalmente se creó un prototipo que con solo una imagen de la muestra reconoce y clasifica el tipo de contaminante presente; desplegando el modelo de aprendizaje automático obtenido en una tarjeta Raspberry Pi 4 de 4 GB, a fin de que pueda ser usado como un clasificador de agentes contaminantes.

Seguidas todas las pautas de este proyecto, y una vez completado el detector y clasificador utilizando el modelo arquitectura ResNet-50 V2 (opción de precisión en Lobe), permiten llegar a una solución vigorosa, en cuanto a la detección y clasificación de los agentes contaminantes presentes en la superficie de la muestra de agua.

Ventajas

- Lobe es una aplicación de escritorio gratuita, que permite a los usuarios sin mayores conocimientos en programación o ciencias de datos, agregar capacidades de aprendizaje automático a sus aplicaciones. De hecho, realiza todo el proceso para la obtención de un modelo de aprendizaje automático en forma gráfica, posee un entorno bastante intuitivo y fácil de usar; sin la necesidad de escribir una sola línea de código.

- Lobe es una herramienta eficaz que no compromete la privacidad del usuario y sus datos. La aplicación funciona sin conexión y todos los datos importados permanecen en la computadora, en lugar de cargarse en la nube (y Microsoft).

- El entrenamiento del modelo empieza automáticamente mientras las imágenes son cargadas, una vez se cumplan que mínimo existan 2 categorías con al menos cinco imágenes por cada una de ellas. Además un nuevo modelo es desarrollado inmediatamente; si nuevas imágenes son cargadas o se cambia las etiquetas de las ya ingresadas, sin la necesidad de realizar ninguna acción adicional.

- El uso de la Raspberry Pi 4 permite la integración de nuevas herramientas tecnológicas orientadas al desarrollo de proyectos de IA con las librerías de Lobe y Phyton.

- El protocolo MIPI es altamente eficiente y confiable puede manejar videos de 1080 a 8K y más, tiene un ancho de banda alto de 6 Gb/s con cuatro líneas de datos de imágenes, cada una con capacidad de 1,5 Gb/s; esto hace que tenga una baja sobrecarga y un mayor ancho de banda neto. Por ende MIPI CSI-2 es más rápido que que la interfase multipropósito USB 3.0 que puede llegar a 5 Gb/s.[[Vision-Online-Marketing-Team, 2020](#)]

- En este trabajo se presenta un nuevo enfoque a los sistemas de detección y clasificación tradicionales (usados actualmente); además da una alternativa innovadora para la determinación de la calidad y supervisión de un espejo agua, ya que utilizar técnicas de aprendizaje profundo (Deep Learning), representa una gran ventaja sobre las acciones de extracción de características y clasificación de imágenes, sabiendo que las CNN (Convolutional Neural Network) manejan grandes cantidades de datos y son capaces de tomar determinaciones o acciones de manera casi instantánea

con eficiencia y eficacia.

El modelo hallado puede ser probado para predecir el tipo de contaminante usando nuevas imágenes desde la computadora, obteniendo los resultados de manera inmediata; pudiendo determinar así, los patrones recurrentes en las imágenes incorrectas. Incluso de considerarlo conveniente incluir las imágenes que engañan al modelo con la etiqueta correcta a la base de datos del entrenamiento.

Desventajas

No se puede usar videos directamente con Lobe, en caso de requerirlo para recopilar imágenes hay que convertir el video en cuadros y seleccionar la mejor y más alta variedad de imágenes para el problema. Debiendo utilizar herramientas como SnapMotion, VLC, Free Video to JPG Converter o VirtualDub.

El entrenamiento es un proceso de computación muy pesado (demandante de recursos computacionales). Lobe utilizará mucha memoria y ancho de banda de CPU para entrenar lo más rápido posible. Esto generalmente significa que el computador tendrá menos CPU y memoria disponible para que otras aplicaciones la usen.

Recomendaciones

Para la captura de las imágenes de muestras de agua con la cámara; se debe tener en cuenta, que la iluminación natural existente sea la apropiada (todas las capturas fueron realizadas con luz natural durante el día); ya que una excesiva iluminación, producirá el efecto indeseable de la reflexión directa; en cambio una iluminación deficiente no permitirá distinguir los patrones propios de la muestra; en ambos casos se provocaría pérdidas de características de la muestra, provocando ruido y confundiendo al modelo.

Para trabajar con redes neuronales de gran tamaño (muchas capas ocultas) y con base de datos de tamaño considerable (cientos o miles de imágenes por etiqueta), es recomendable disponer de un procesador con características apropiadas para el gestionamiento de las imágenes, incluso evaluar el uso de GPU's independientes para entrenar la red en tiempos relativamente razonables.

Al analizar los resultados se debe tener en cuenta que si el rendimiento general en todas las imágenes es muy alto y el rendimiento con las imágenes de prueba es bajo, es posible que el modelo esté memorizando los ejemplos en lugar de aprender a generalizar a imágenes no vistas. A esto se le suele llamar sobreajuste. Para ayudar a prevenir el sobreajuste, se puede: a) Recopilar más imágenes en general para que el conjunto de entrenamiento incluya más variaciones y así eliminar cualquier tipo de sesgo. b) Asegurarse de que las imágenes no se parezcan demasiado entre sí.

Se debe considerar utilizar el modelo de optimizado por velocidad; si se necesita baja latencia (total de retardos temporales dentro de una red) y un menor uso de memoria.

Lobe siempre predecirá una de sus etiquetas suministradas para cada imagen. Si espera que el modelo vea imágenes que no pertenecen a ninguna de las etiquetas deseadas, se puede crear una etiqueta “Ninguno” como un depósito general para mostrar imágenes no relacionadas.

Sobre las imágenes a emplear como parte del juego de datos, de manera general se puede aconsejar lo siguiente:

Recopilar imágenes que se espera ver en el mundo real y capturar tantas variaciones como sea posible; esto es reunir tantas imágenes diferentes como pueda.

Intentar capturar todas las variaciones que ocurren naturalmente mediante la recopilación de imágenes en diferentes condiciones. Probar con diferentes fondos, iluminación, orientaciones o zoom. Esto ayuda a Lobe a saber que partes de la imagen son útiles para hacer predicciones y que es ruido.

Cabe anotar que Lobe solo puede aprender los patrones que existen en las imágenes que se le proporcionan como ejemplos. Por esto se debe recopilar las imágenes de la misma fuente que espera usar para el modelo exportado.

Asegurarse de que el contenido de la muestra a evaluar sea visible en el cuadrado central de la imagen, ya que mientras entrena el modelo, Lobe recorta el cuadrado central de sus imágenes.

Cuanto más pequeño sea el objeto que se intenta clasificar, más difícil será para el modelo predecirlo; una solución de ser posible, sería configurar la cámara para que esté más cerca o amplíe el objeto que se necesita clasificar. Alternativamente, se puede recortar el cuadrado de la parte de la imagen que contiene el objeto que desea clasificar.

Una pauta aproximada para el número de imágenes necesarias para la obtención de un modelo, es de 100 a 1000 imágenes por etiqueta para problemas más pequeños.

Tener aproximadamente el mismo número de imágenes por etiqueta. Para equilibrar el número de imágenes entre cada etiqueta. Si está desequilibrado, Lobe estará sesgado para predecir las etiquetas con más imágenes e ignorar las etiquetas con menos imágenes.

Apéndice A

Programa completo del Clasificador

A continuación se describe de manera integral el programa desarrollado en Python que fue usado para el desarrollo de este proyecto.

```
#import Pi GPIO library button class
from gpiozero import Button, LED, PWMLED
from picamera import PiCamera
from time import sleep
from lobe import ImageModel

#Create input, output, and camera objects
button = Button(2)
yellow_led = LED(17) #Aceite
blue_led = LED(27) #Potable
orange_led = LED(22) #Sólidos
red_led = LED(23) #Turbia
white_led = PWMLED(24) #Status light and retake photo
power_led = PWMLED(25) #Indicador ON/OFF(green)
camera = PiCamera()

# Load Lobe TF model
# ->Change model file path as needed
model = ImageModel.load('/home/pi/Lobe/modeloCont')
```

```

# Take Photo
def take_photo():

    # Start the camera preview
    camera.start_preview(alpha=255)
    # wait 2s or more for light adjustment
    sleep(3)
    # Optional image rotation for camera
    # ->Change or comment out as needed
    camera.rotation = 0
    #Input image file path here
    # ->Change image path as needed
    camera.capture('/home/pi/Pictures/image.jpg')
    #Stop camera
    camera.stop_preview()
    white_led.off()

# Identify prediction and turn on appropriate LED
def led_select(label):

    print(label)
    if label == "Aceite":
        yellow_led.on()
        sleep(5)
    if label == "Con Sólidos":
        orange_led.on()
        sleep(5)
    if label == "Potable":
        blue_led.on()
        sleep(5)
    if label == "Turbia":
        red_led.on()
        sleep(5)
        red_led.off()
    else:
        yellow_led.off()
        blue_led.off()
        orange_led.off()
        red_led.off()
        white_led.off()

```

```
# Main Function
power_led.on()
while True:

    if button.is_pressed:
        white_led.pulse(0.5,0.5)
        take_photo()
        # Run photo through Lobe TF model
        sleep(0.5)
        white_led.pulse(0.1,0.1)
        result = model.predict_from_file('/home/pi/Pictures/image.jpg')
        white_led.off()
        led_select(result.prediction)
    sleep(1)
```

Bibliografía

- J. Bagnato. Convolutional neural networks: La teoría explicada en español | aprende machine learning, 11 2018. URL <https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador>.
- J. I. Barrios. La matriz de confusión y sus métricas – inteligencia artificial –, Aug 2021. URL <https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/>.
- J. Cabot. Crea aplicaciones de inteligencia artificial sin programar, 12 2018. URL <https://ingeneriadesoftware.es/crea-aplicaciones-de-inteligencia-artificial-sin-programar/>.
- S. R. Carvalho, I. C. Filho, D. O. D. Resende, A. C. Siravenha, C. R. De Souza, H. Debarba, B. D. Gomes, and R. Boulic. A deep learning approach for classification of reaching targets from eeg images. In *2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 178–184, 2017. doi: 10.1109/SIBGRAPI.2017.30.
- Cognex Corporation. *Introducción a la Visión artificial*. Cognex, USA, 2016.
- DeepAI. Computer vision, 05 2019a. URL <https://deepai.org/machine-learning-glossary-and-terms/computer-vision>.
- DeepAI. Machine learning, 05 2019b. URL <https://deepai.org/machine-learning-glossary-and-terms/machine-learning>.
- F. Deirdre. Computer vision is a field that includes methods for acquiring, processing, analyzing, and understanding images and, in general, high-dimensional data. - ppt download, 2016. URL <https://slideplayer.com/slide/10117716/>.

- Y.-S. Deng, A.-C. Luo, and M.-J. Dai. Building an automatic defect verification system using deep neural network for pcb defect classification. In *2018 4th International Conference on Frontiers of Signal Processing (ICFSP)*, pages 145–149, 2018. doi: 10.1109/ICFSP.2018.8552045.
- H. Ding, R. R. Li, H. Lin, and X. Wang. Monitoring and evaluation on water quality of hun river based on landsat satellite data. In *2016 Progress in Electromagnetic Research Symposium (PIERS)*, pages 1532–1537, 2016. doi: 10.1109/PIERS.2016.7734699.
- Ecuavisa-Redacción. Personal de petroecuador detecta otra amenaza en el río daule, Jun 2016. URL <https://www.ecuavisa.com/noticias/ecuador/personal-petroecuador-detecta-otra-amenaza-rio-daule-BBEC168081>. Online; accessed 17 January 2020.
- ElComercio-Redacción-Guayaquil. Derrame de búnker en río daule alcanzó los 20 kilómetros, Jun 2016. URL <https://www.elcomercio.com/actualidad/ecuador/derrame-bunker-rio-daule-guayaquil.html>. Online; accessed 17 January 2020.
- Foundation-Raspberry. Raspberry pi os. URL <https://www.raspberrypi.org/software/>.
- L. García. Aprendizaje profundo para visión artificial con matlab video, 2016. URL <https://la.mathworks.com/videos/deep-learning-for-computer-vision-with-matlab-1540981496452.html>.
- B. G. Gautama, N. Longépé, R. Fablet, and G. Mercier. Assimilative 2-d lagrangian transport model for the estimation of oil leakage parameters from sar images: Application to the montara oil spill. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(11): 4962–4969, 2016. doi: 10.1109/JSTARS.2016.2606110.
- Google-Developers. Classification: Roc curve and auc | machine learning crash course, 2019. URL <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>.
- K. Gopavanitha and S. Nagaraju. A low cost system for real time water quality monitoring and controlling using iot. In *2017 International conference on energy, communication, data analytics and soft computing (ICECDS)*, pages 3227–3229. IEEE, 2017.

- D. J. Hand and R. J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2): 171–186, 2001. doi: 10.1023/a:1010920819831.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.
- A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.
- E. M. i García. Visión artificial. *Fundación para la universidad Oberta de Catalunya*, 2012.
- Industries-Adafruit. Raspberry pi camera board. URL <https://www.adafruit.com/product/1367>.
- INEC-Censos. Ecuador cuenta con su reloj poblacional. URL <https://www.ecuadorencifras.gob.ec/ecuador-cuenta-con-su-reloj-poblacional/>.
- Isaac. Gpio: todo sobre las conexiones de la raspberry pi 4 y 3, 04 2020. URL <https://www.hwlibre.com/gpio-raspberry-pi/>.
- P. Kamsing, P. Torteeka, and S. Yooyen. Deep convolutional neural networks for plane identification on satellite imagery by exploiting transfer learning with a different optimizer. In *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 9788–9791, 2019. doi: 10.1109/IGARSS.2019.8899206.
- A. Kordon. *Applying computational intelligence how to create value*. Springer Berlin Heidelberg, 2010.
- G. Li, Y. Li, Y. Hou, and T. Wang. Analysis of oil spill properties based on dual-polarization radarsat-2 imagery. In *2017 SAR in Big Data Era: Models, Methods and Applications (BIGSAR DATA)*, pages 1–4, 2017. doi: 10.1109/BIGSAR DATA.2017.8124920.
- J. M. Malof, L. M. Collins, and K. Bradbury. A deep convolutional neural network, with pre-training, for solar photovoltaic array detection in aerial imagery. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 874–877, 2017. doi: 10.1109/IGARSS.2017.8127092.

- Mathworks. Introducing deep learning with matlab, 2018. URL https://la.mathworks.com/content/dam/mathworks/ebook/gated/80879v00_Deep_Learning_ebook.pdf.
- Mathworks. Machine learning with matlab, 2020. URL <https://la.mathworks.com/content/dam/mathworks/ebook/gated/machine-learning-ebook-all-chapters.pdf>.
- N. Mhaisen, O. Abazeed, Y. Al Hariri, A. Alsalemi, and O. Halabi. Self-powered iot-enabled water monitoring system. In *2018 International Conference on Computer and Applications (ICCA)*, pages 41–45. IEEE, 2018.
- Ministerio de Ambiente. Programa ‘pngids’ ecuador – ministerio del ambiente, agua y transición ecológica. URL <https://www.ambiente.gob.ec/programa-pngids-ecuador/>.
- V. S. Mohan, V. Sowmya, and K. P. Soman. Deep neural networks as feature extractors for classification of vehicles in aerial imagery. In *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 105–110, 2018. doi: 10.1109/SPIN.2018.8474153.
- J. Ordieres and F. Martínez. Técnicas y algoritmos básicos de visión artificial, 01 2006. URL https://www.researchgate.net/publication/231521316_Tecnicas_y_algoritmos_basicos_de_vision_artificial_Recurso_electronico_-_En_linea.
- I. C. y. I. C. Organización de las Naciones Unidas para la Educación. Agua y empleo informe de las naciones unidas sobre el desarrollo de los recursos hídricos en el mundo 2016, 2016. URL <https://unesdoc.unesco.org/ark:/48223/pf0000244103>.
- J. Palma and R. Marín. *Inteligencia Artificial: Técnicas, métodos y aplicaciones*. McGraw Hill, 2008.
- P. Ponce and A. Herrera. *Inteligencia artificial con aplicaciones a la ingeniería*. Alfaomega Grupo Editor, 2010.
- M. B. Praba, N. Rengaswamy, O. Deepak, et al. Iot based smart water system. In *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, pages 1041–1045. IEEE, 2018.
- F. Provost and P. Domingos. Well-trained pets: Improving probability estimation trees, 2000.

- C. Rajurkar, S. Prabaharan, and S. Muthulakshmi. Iot based water management. In *2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2)*, pages 255–259. IEEE, 2017.
- raspberrypi.org, 2017. URL <https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/2>.
- Raspberrypi-Projects, 2017. URL <https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/3>.
- M. Rivera. resnet.md, 08 2019. URL http://personal.cimat.mx:8181/~mriviera/cursos/aprendizaje_profundo/resnet/resnet.html.
- L. Rouhiaien. *Inteligencia artificial: 101 cosas que debes saber hoy sobre nuestro futuro*. Alienta Editorial, Barcelona, 2018.
- M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. doi: 10.1109/CVPR.2018.00474.
- Z. Shanhua, P. Li, and F. Ning-Sheng. Real-time detection method of surface floating objects based on deep learning. In *2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 174–177, 2020. doi: 10.1109/DCABES50732.2020.00053.
- R. Shanmugamani. *Deep learning for computer vision : expert techniques to train advanced neural networks using TensorFlow and Keras*. Packt Publishing, 2018.
- M. Suresh, U. Muthukumar, and J. Chandapillai. A novel smart water-meter based on iot and smartphone app for city distribution management. In *2017 IEEE region 10 symposium (TENSYP)*, pages 1–5. IEEE, 2017.
- E. Valveny. Evaluación del rendimiento - introducción a la clasificación de imágenes. URL <https://es.coursera.org/lecture/clasificacion-imagenes/evaluacion-del-rendimiento-9YvpP>.
- Vision-Online-Marketing-Team. Comparing the different interface standards for embedded vision, 2020. URL <https://www.automate.org/blogs/comparing-the-different-interface-standards-for-embedded-vision>.

- J. Wang, J. Zhang, T. Li, and X. Wang. Water quality analysis of remote sensing images based on inversion model. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 4861–4864, 2018. doi: 10.1109/IGARSS.2018.8519442.
- Y. Xu, Y. Feng, Z. Xie, A. Hu, and X. Zhang. A research on extracting road network from high resolution remote sensing imagery. In *2018 26th International Conference on Geoinformatics*, pages 1–4. IEEE, 2018.
- H. L. Yang, J. Yuan, D. Lunga, M. Laverdiere, A. Rose, and B. Bhaduri. Building extraction at scale using convolutional neural network: Mapping of the united states. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(8):2600–2614, 2018. doi: 10.1109/JSTARS.2018.2835377.
- J. Yang, J. Wan, Y. Ma, and Y. Hu. Research on object-oriented decision fusion for oil spill detection on sea surface. In *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 9772–9775, 2019. doi: 10.1109/IGARSS.2019.8899010.