SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

INSTITUTO TECNOLÓGICO DE MORELIA

Nº 084

ACTA DE EXAMEN DE GRADO DE MAESTRÍA

En la ciudad de Morelia, Michoacán, a los 06 días del mes de Junio, de 2022, siendo las 11:00 horas; se reunieron en Sala Audiovisual del Instituto Tecnológico de Morelia clave 16DIT0012H, el jurado integrado por:

Presidente (a): Doctor en Tecnología de Información y Análisis de Decisiones, Heberto Ferreira Medina, 9875495

Secretario (a): Doctor en Ciencias de la Ingeniería, Juan Carlos Olivares Rojas, 12738766

Vocal: Doctor en Ciencia de la Ingeniería Eléctrica Opción en Sistemas Computacionales

Y de acuerdo con las disposiciones reglamentarias en vigor, se procedió a efectuar el examen de Grado de Maestría a el (la) C. Ulises Martínez Covarrubias número de control M10011012, aspirante al Grado de Maestría en Sistemas Computacionales

Tomando en cuenta el contenido de la tesis cuyo título es: Diseño e implementación de un Sistema de Detección de Mosquitos Basado en Patrones; Creación de la Plataforma Web y Servicios Seguros

que fue dirigida por Heberto Ferreira Medina, una vez oído el examen oral, dictaminó que fuera Aprobado

El (la) Presidente (a) del jurado le hizo saber a el (la) sustentante el resultado obtenido, el Código Profesional y le tomó la Protesta de Ley. Dándose por terminado el Acto a las 12:00 horas, escrita, leída y aprobada, fue firmada para constancia por las personas que en el acto intervinieron efectos legales a que haya lugar se asienta la presente.

SECRETARÍA DE ESTADOS TECNOLÓGICOS DE M... DIRECCIÓN DE ASI... Y APOYO A...

PRESIDENTE (A)

Heberto Ferreira Medina

Doctor en Tecnología de Información y Análisis de Decisiones

Cédula de Grado 9875495

SECRETARIO (A)

Juan Carlos Olivares

Juan Carlos Olivares Rojas

Doctor en Ciencias de la Ingeniería

Cédula de Grado 12738766

VOCAL

Anastacio Antolino Hern...

Doctor en Ciencias en Ingenier... Opción en Sistemas C...

Cédula de Grado 7355312

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Morelia
Subdirección Académica
División de Estudios de Posgrado e Investigación
Coordinación de la Maestría en Sistemas Computacionales

Morelia, Michoacán, 27/Noviembre/2021

OFICIO No. 014/2021

ACTA DE REVISIÓN DE TESIS

En la ciudad de Morelia, Michoacán, siendo las 11:00 horas del día 26 de Noviembre del 2021, se reunieron los Miembros de la Comisión Tutorial de la Tesis designada por el H. Consejo de Posgrado de la Maestría en Sistemas Computacionales, para examinar la tesis de grado titulada:

"Diseño e implementación de un sistema de detección de mosquitos basado en patrones; creación de la plataforma Web y servicios seguros"

Presentada por el alumno:

ULISES MARTÍNEZ COVARRUBIAS, con número de control M10011012

Aspirante al grado de:

MAESTRO EN SISTEMAS COMPUTACIONALES

Después de intercambiar opiniones los miembros de la comisión manifestaron su aprobación para la impresión final de la tesis, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

Presidente y Director de Tesis:

Heberto Ferreira Medina
Doctor en Tecnologías de Información y Análisis de Decisiones
No. De Cédula Profesional: 9875495

Secretario: Juan Carlos Olivares Rojas (30 Nov. 2021 08:25 CST)

Juan Carlos Olivares Rojas
Maestro en Ciencias en Ciencias de la Computación
No. de Cédula Profesional: 5085191

Vocal: Anastacio Antolino Hernández (30 Nov. 2021 08:05 CST)

Anastacio Antolino Hernández
Doctor en Ciencias en Ingeniería Eléctrica Opción en Sistemas Computacionales
No. de Cédula Profesional: 7355312

Vocal Suplente:

Abel Alberto Pintor Estrada
Maestro en Ciencias en Ciencias de la Computación
No. de Cédula Profesional: 10228740

Vo.Bo. Miriam Zulma Sanchez (1 Dec. 2021 08:44 CST)

Miriam Zulma Sánchez Hernández
Presidenta del Consejo de Posgrado

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE MORELIA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE MORELIA
"José María Morelos y Pavón"

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE MORELIA

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACION

TESIS

"Diseño e implementación de un sistema de detección de mosquitos basado en patrones; creación de la plataforma Web y servicios seguros"

QUE PARA OBTENER EL TÍTULO DE:
MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:
ING. ULISES MARTÍNEZ COVARRUBIAS

DIRECTOR DE TESIS:
DR. HEBERTO FERREIRA MEDINA

MORELIA, MICHOACÁN NOVIEMBRE DE 2021

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE DETECCIÓN DE MOSQUITOS BASADO EN PATRONES; CREACIÓN DE LA PLATAFORMA WEB Y SERVICIOS SEGURO.

Instituto Tecnológico de Morelia
Subdirección Académica
División de Estudios de Posgrado e Investigación
Coordinación de la Maestría en Sistemas Computacionales

Morelia, Michoacán, 27/Noviembre/2021

OFICIO No. 014/2021

ACTA DE REVISIÓN DE TESIS

En la ciudad de Morelia, Michoacán, siendo las 11:00 horas del día 26 de Noviembre del 2021, se reunieron los Miembros de la Comisión Tutorial de la Tesis designada por el H. Consejo de Posgrado de la Maestría en Sistemas Computacionales, para examinar la tesis de grado titulada:

"Diseño e implementación de un sistema de detección de mosquitos basado en patrones; creación de la plataforma Web y servicios seguros"

Presentada por el alumno:

ULISES MARTÍNEZ COVARRUBIAS, con número de control M10011012

Aspirante al grado de:

MAESTRO EN SISTEMAS COMPUTACIONALES

Después de intercambiar opiniones los miembros de la comisión manifestaron su aprobación para la impresión final de la tesis, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

Presidente y Director de Tesis:

Heberto Ferreira Medina
Doctor en Tecnologías de Información y Análisis de Decisiones
No. De Cédula Profesional: 9875495

Secretario:

Juan Carlos Olivares Rojas (30 Nov. 2021 08:25 CST)

Juan Carlos Olivares Rojas
Maestro en Ciencias en Ciencias de la Computación
No. de Cédula Profesional: 5085191

Vocal:

Anastacio Antolino Hernández (30 Nov. 2021 08:05 CST)

Anastacio Antolino Hernández
Doctor en Ciencias en Ingeniería Eléctrica
Opción en Sistemas Computacionales
No. de Cédula Profesional: 7355312

Vocal Suplente:

Abel Alberto Pintor Estrada
Maestro en Ciencias en Ciencias de la Computación
No. de Cédula Profesional: 10228740

Vo.Bo.

Miriam Zulma Sanchez Hernandez (1 Dec. 2021 08:40 CST)

Miriam Zulma Sánchez Hernández
Presidenta del Consejo de Posgrado

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE MORELIA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Morelia
Subdirección Académica
División de Estudios de Posgrado e Investigación

Morelia, Michoacán, 25/ABRIL/2022

OFICIO No. DEPI/108/2022

ASUNTO: Autorización de impresión definitiva de tesis

ULISES MARTÍNEZ COVARRUBIAS
EGRESADO DE LA MAESTRÍA EN
SISTEMAS COMPUTACIONALES
PRESENTE

Le comunico que el jurado designado para que obtenga el grado de **MAESTRO EN SISTEMAS COMPUTACIONALES**, ha informado a esta División de Estudios de Posgrado e Investigación, su aceptación para la impresión definitiva de su trabajo de tesis, el cual lleva por nombre: **"Diseño e implementación de un sistema de detección de mosquitos basado en patrones; creación de la plataforma Web y servicios seguros"**, Por lo anterior se le autoriza la impresión de su trabajo, esperando que el logro del mismo sea acorde con sus aspiraciones profesionales.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Técnica, Progreso de México"®

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE MORELIA
DIRECCION

HÉCTOR JAVIER VERGARA HERNÁNDEZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

C.p. Archivo
HJVH/lsa*

Recibí Estudiante
18/05/22

2022 Ricardo Flores Magón

AGRADECIMIENTOS

Agradezco primeramente a Dios por darme la vida y a mi familia por todo el apoyo en cada decisión y proyecto.

Agradezco todo el apoyo a este proyecto del Tecnológico Nacional de México Campus Morelia mediante el proyecto TecNM 2021, con clave 10614.21-P “Detección de anomalías de seguridad en redes de domótica, utilizando Machine Learning” todo su apoyo. El apoyo del IIES-UNAM y al proyecto PAPIME de la DGAPA PE106021 ”Propuesta de mejora a la enseñanza del aprendizaje automático aplicado a la Ciencia de Datos a gran escala; dirigido a académicos y estudiantes de la licenciatura en Tecnologías para la Información en Ciencias (LTICs) en la ENES Morelia.”, al Dr. Sergio Tinoco Martínez, MTI. Froylan Hernández Rendón de la ENES Morelia, al Dr. José L. Cendejas Valdez de la UTM, por su apoyo técnico. Agradezco también a los profesores del IIES-UNAM, que apoyaron en el desarrollo del prototipo y la página Web; MGTI. Atzimba López M., MTI. Alberto Valencia G., y al MGTI. Gustavo A. Vanegas, al Ing. Marco Rojas de la UTM y a los estudiantes de la UTM C. Cristina Juárez y C. Rodrigo Pasaye.

ÍNDICE

Índice de Figuras

Índice de Tablas

Resumen

En este trabajo se analizan los patrones de imágenes de mosquito adquiridas desde una aplicación móvil, para detectar la presencia de mosquitos *Aedes aegypti* y *Aedes albopictus* en poblaciones urbanas, que generan un problema de salud al convertirse en vectores de las enfermedades del Dengue, Zika y Chikungunya. El proyecto consiste en el desarrollo de un sistema de software REST API basado en una aplicación Web y el lenguaje Python que implementa redes neuronales convolutivas (CNN) para el Deep Learning (DL), mediante el uso de la biblioteca TensorFlow y Keras. Se utiliza un servidor Web seguro con el framework Django, el cual implementa la comunicación entre la aplicación móvil y el sistema de DL. Las funciones del servicio Web son: 1) publicar el servicio para enviar imágenes, 2) almacenarlas y clasificarlas y 3) analizar la similitud con los mosquitos. Los resultados son guardados en una base datos para su posterior consulta a través del sistema móvil o Web. Se proporciona el informe de los análisis y el porcentaje de similitud de imagen probada. La metodología utilizada en este proyecto sigue las fases; 1) almacenamiento de imágenes, 2) tratamiento utilizando DL, 3) uso de CNN, 4) uso del modelo de aprendizaje para establecer la similitud y 5) reporte de resultados. Los resultados del sistema permiten garantizar que la tecnología desarrollada es confiable en la detección de mosquitos *Aedes aegypti* y *Aedes albopictus* (porcentajes de detección alcanzados en el sistema de entre 60% y 95%), por lo que su potencial de aplicación en la industrial y sector salud es viable.

Abstract

In this work, the patterns of images of mosquitoes acquired through a mobile application are analyzed to detect the presence of *Aedes aegypti* and *Aedes albopictus* mosquitoes in urban populations, which generate a health problem, by becoming vectors of diseases; Dengue, Zika and, Chikungunya. A REST API software system is developed based on a Web application and the Python language that implements convolutional neural networks (CNN) for Deep Learning (DL) using the TensorFlow library and Keras. A secure Web server Django framework is used, which implements the communication between the mobile application and the DL system. The functions of the Web service are: 1) publish the service to send images, 2) store and classify them and 3) analyze the similarity to mosquitoes. The results are stored in a database for later consultation through the mobile system or the Web. The analysis report and tested image similarity percentage are provided. The methodology utilized in this project follows the phases; 1) image storage, 2) treatment using DL, 3) use of CNN, 4) use of the learning model to establish similarity, and 5) report of results. The results of the system make it possible to guarantee that the technology developed is reliable in detecting *Aedes aegypti* and *Aedes albopictus* mosquitoes (detection percentages achieved in the system between 60% and 95%), so that its application potential in the industrial and health sectors is viable.

Capítulo 1. Introducción

En este capítulo se describe el planteamiento del problema, así como los objetivos, alcances y limitaciones del proyecto. Dando así una introducción a todo el proyecto.

1.1. Planteamiento del problema

El dengue es una enfermedad causada por un virus y que su vector transmisión es por diferentes especies de mosquitos que se han propagado rápidamente en muchas regiones del planeta y que la OMS reconoce como un problema de salud en los últimos años. El virus del dengue se transmite por mosquitos hembra principalmente de la especie *Aedes aegypti* y en menor grado por el *Aedes albopictus*. Estos mosquitos también transmiten la fiebre Chikungunya, la fiebre amarilla y la infección del Zika. Estas enfermedades están muy extendidas en los países tropicales, con variaciones locales en su incidencia que dependen en gran medida de la precipitación pluvial, la temperatura y la urbanización rápida sin planificar (Bhatt, Gething, Brady, Messina, Farlow, & Moyes, 2019).

El dengue grave (conocido también como hemorrágico) fue identificado desde los años cincuenta durante una epidemia en Filipinas y Tailandia. Hoy en día, afecta a la mayor parte de los países de Asia, América Latina y Europa y se ha convertido en una de las causas principales de hospitalización y muerte en los niños y adultos en muchas regiones (Kourí, 2006).

El causante del dengue es un virus de la familia Flaviviridae que tiene cuatro serotipos distintos estrechamente emparentados: DEN-1, DEN-2, DEN-3 y DEN-4. Cuando una persona se recupera de la infección adquiere inmunidad de por vida contra el serotipo en particular. Las infecciones posteriores (secundarias) causadas por otros serotipos aumentan el riesgo de padecer el dengue grave. El vector principal del dengue es el mosquito *Aedes aegypti*. El virus se transmite a los seres humanos por la picadura de mosquitos hembra infectadas. Tras un periodo de incubación del virus que dura entre 4 y 10 días, un mosquito infectado puede transmitir el agente patógeno durante su ciclo de vida.

Las personas infectadas sintomáticas y asintomáticas son los portadores y multiplicadores principales del virus y los mosquitos se convierten en portadores al picarlas.

Tras la aparición de los primeros síntomas, las personas infectadas pueden transmitir el virus (durante 4 a 12 días como máximo) a los mosquitos. El *Aedes aegypti* vive en hábitats urbanos y se reproduce principalmente en recipientes artificiales. A diferencia de otros mosquitos, este se alimenta durante el día; los periodos en que se intensifican las picaduras son al principio de la mañana y en el atardecer, antes de que oscurezca. En cada periodo de alimentación, el mosquito hembra puede llegar a picar a muchas personas. Los huevos de *Aedes aegypti* pueden permanecer secos en sus lugares de cría durante más de un año y eclosionar al entrar en contacto con el agua.

Aedes albopictus, vector secundario del dengue en Asia, se ha propagado a Canadá, Norte América y a más de 25 países en la región de Europa debido al comercio internacional de neumáticos usados (que proporcionan criaderos al mosquito) y el movimiento de mercancías (por ejemplo, el bambú de la suerte). *Aedes albopictus* tiene una gran capacidad de adaptación y gracias a ello puede sobrevivir en las temperaturas más frías de Europa. Su tolerancia a las temperaturas bajo cero, su capacidad de hibernación y su habilidad para guarecerse en micro hábitats son factores que propician su propagación (Kourí, 2006).

El dengue es una enfermedad de tipo gripal que afecta a bebés, niños pequeños y adultos, pero raras veces resulta mortal. Se debe sospechar que una persona padece dengue cuando una fiebre elevada (40 °C) se acompaña de al menos dos de los síntomas siguientes: dolor de cabeza muy intenso, dolor detrás de los globos oculares, dolores musculares y articulares, náuseas, vómitos, agrandamiento de ganglios linfáticos o salpullido. Los síntomas se presentan al cabo de un periodo de incubación de 4 a 10 días después de la picadura de un mosquito infectado y por lo común duran entre 2 y 7 días.

El dengue grave es una complicación potencialmente mortal porque cursa con extravasación de plasma, acumulación de líquidos, dificultad respiratoria, hemorragias graves o falla orgánica. Los signos que advierten de esta complicación se presentan entre 3 y 7 días después de los primeros síntomas y se acompañan de un descenso de la temperatura corporal (menos de 38 °C) y son los siguientes: dolor abdominal intenso, vómitos persistentes, respiración acelerada, hemorragias de las encías, fatiga, inquietud y presencia de sangre en el vómito. Las siguientes 24 a 48 horas de la etapa crítica pueden ser letales;

hay que brindar atención médica para evitar otras complicaciones y disminuir el riesgo de muerte (Bhatt, Gething, Brady, Messina, Farlow, & Moyes, 2019).

Una de las estrategias más comunes para controlar o erradicar a los mosquitos del género Aedes, es la realización de campañas de fumigación con insecticidas y la eliminación de sitios donde las hembras mosquito ponen sus huevecillos ("deschatarrización"). Debido a que la distribución y abundancia de los mosquitos es heterogénea y puede cambiar de temporada a temporada, es indispensable detectar la abundancia y distribución de los mosquitos para dirigir las campañas con precisión y evitar fumigaciones generalizadas.

Dado que el dengue es uno de los principales problemas de salud pública en América Latina, con morbi-mortalidad creciente en los últimos 30 años. En el año 2010 se tiene el reporte más alto de casos en el continente americano en el año 2019 nuevamente repuntó. Ante esta situación se está tratando de poner en el centro de la estrategia el control integrado del vector, lo que deja al descubierto las debilidades de varios de los componentes necesarios para llevarla a cabo. En este marco se destacó la necesidad de impulsar la innovación en el control de vectores, e incrementar recursos humanos capacitados y materiales para dar respuesta necesaria, adecuada y oportuna (Kourí, 2006).

1.2. Objetivo

Desarrollar un sistema de detección de mosquitos transmisores *Aedes aegypti* y *Aedes albopictus* mediante el análisis de patrones usando redes neuronales convolutivas en imágenes, para detectar la propagación de enfermedades transmitidas por los mosquitos a la población humana.

1.2.1. Objetivos específicos

1. Construir una plataforma Web que permita el análisis y detección de mosquitos trasmisores de enfermedades utilizando herramientas de Deep Learnig.
2. Procesar las imágenes de mosquitos enviadas a través de un servicio Web.
3. Analizar patrones en las imágenes para determinar si se trata de mosquitos transmisores *Aedes aegypti* y *Aedes albopictus*.

4. Implementar aprendizaje profundo (Deep Learning) mediante la librería TensorFlow en un sistema desarrollado en Python.
5. Proporcionar los datos obtenidos por el sistema encargado de detectar mosquitos transmisores para que sean evaluados.
6. Desarrollar un servicio Web seguro para establecer comunicación con una aplicación móvil encargada de obtener las imágenes digitales de los mosquitos.

1.3. Hipótesis

La implementación de un sistema de software basado en Deep Learning (CNN), para el análisis de patrones de mosquitos transmisores en imágenes tomadas con un sistema de adquisición móvil, permite la detección rápida de vectores de virus *Aedes aegypti* y *Aedes albopictus* en zonas urbanas monitoreadas.

1.4. Alcances

Este proyecto consiste en desarrollar un sistema de software en Python que implemente redes neuronales convolutivas y Deep Learning mediante el uso de la librería TensorFlow. Este sistema se encarga de analizar patrones en imágenes de mosquitos tomadas por una aplicación móvil externa, para detectar la presencia de mosquitos *Aedes aegypti* y *Aedes albopictus*. También se desarrolló un servicio Web seguro mediante el cual existe comunicación entre la aplicación móvil y el sistema encargado de analizar las imágenes. El proyecto consiste en ciertas partes. La primera parte es publicar un servicio Web seguro al cual se puedan enviar las imágenes tomadas por la trampa de mosquitos, este servicio también tiene la función de almacenar dichas imágenes en un servidor de almacenamiento en la nube para que estén accesibles por el sistema encargado de su análisis. Posteriormente se trata la imagen para que esta pueda ser analizada por el sistema encarado del análisis de datos, este sistema emplea redes neuronales convolutivas y Deep Learnig. Una vez terminado el análisis de las imágenes, los resultados son guardados en una base datos para su posterior consulta. Con estos datos otro servicio se encarga de proporcionar los informes del análisis de imágenes, con lo que se expresan los resultados en un porcentaje de similitud.

La metodología a seguir en este proyecto consiste en cinco principales pasos. El primero de ellos es, que una vez adquiridas las imágenes serán almacenadas en un servicio de almacenamiento en la nube. El paso dos es realizar un tratamiento de las imágenes que se encuentran almacenadas para que posteriormente sea posible realizar un análisis profundo de la imagen ya tratada, este análisis es el tercer paso. El cuarto paso consiste en desarrollar un modelo de aprendizaje para detectar si la imagen es correspondiente a los mosquitos transmisores del dengue. Y por último se expondrán los resultados obtenidos del análisis de la imagen. Véase figura 1 de la metodología.

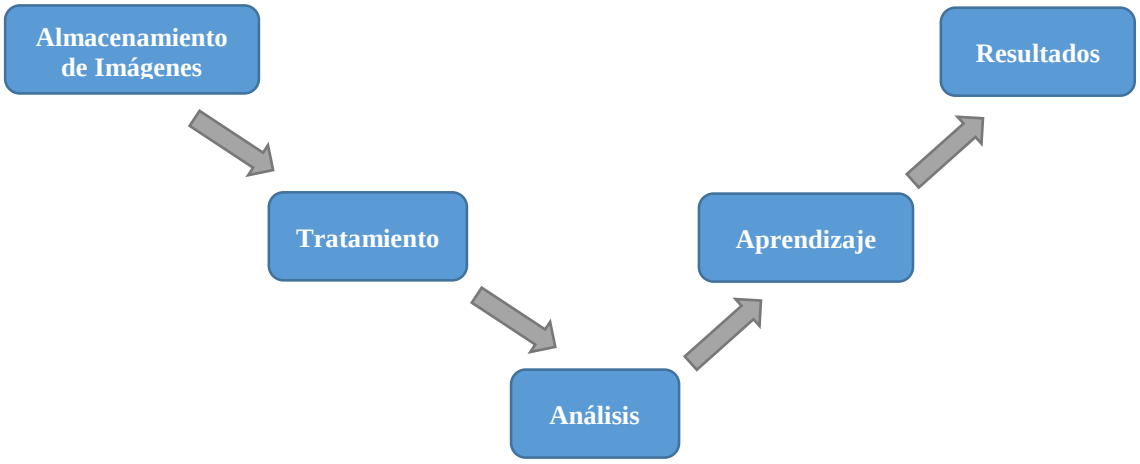

Figura 1. Metodología del proceso para la detección de mosquitos Aedes aegypti y Aedes albopictus

Con el desarrollo de este proyecto, se tiene previsto que se producirá una mejor detección para evitar la propagación del dengue que se producen por la picadura de los mosquitos transmisores, beneficiando así a gran parte de la población humana. Uno de los objetivos de la detección de estos mosquitos es fumigar en vectores de transmisión detectados y evitar fumigaciones generales en las poblaciones para evitar problemas de salud por plaguicidas.

Al tener los conocimientos precisos de si existe o no una población de estos mosquitos en una cierta área, se puede dar paso a tomar medidas para ya sea eliminar estos mosquitos directamente o evacuar primeramente a las personas que puedan estar en esa área localizada

para posteriormente aplicar métodos de la eliminación de los mosquitos transmisores sin afectar a las personas.

1.5. Limitaciones

El sistema encargado de analizar patrones en imágenes para detectar mosquitos transmisores por medio de redes neuronales convolutivas y Deep Learning, se enfocará en comunicarse con la aplicación móvil que tomará las fotos de los mosquitos transmisores, por medio de llamadas a un API REST. Para realizar estas llamadas se desarrollará un WebService programado en Python.

Respecto a la implementación de las redes neuronales convolutivas y el Deep Learning se usó la librería TensorFlow. Esta librería de código abierto se integra adecuadamente con Python y permite la creación de modelos de Deep Learning, desarrollando así modelado predictivo con pocas líneas de código (Brownlee, 2016).

Bibliografía y referencias

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Jia, Y. (9 de Noviembre de 2015). *tensorflow.org.* Obtenido de http://download.tensorflow.org/paper/whitepaper2015.pdf
2. Aguirre Dobernack, N. (s.f.). *Bibing.us.es.* Obtenido de http://bibing.us.es/proyectos/abreproy/12112/fichero/Documento_por_capitulos%252F3_Cap%C3%ADtulo_3.pdf
3. Areli, J., & Barrera, T. (2016). *cucei.udg.mx.* Obtenido de http://www.cucei.udg.mx/sites/default/files/pdf/toral_barrera_jamie_areli.pdf
4. Ayestarán, L. (Febrero de 2018). *Universidad de la Rioja.* Obtenido de https://biblioteca.unirioja.es/tfe_e/TFE003098.pdf
5. Bhatt, S., Gething, P., Brady, O., Messina, J., Farlow, A., & Moyes, C. (04 de Noviembre de 2019). *Organización Mundial de la Salud.* Obtenido de https://www.who.int/es/news-room/fact-sheets/detail/dengue-and-severe-dengue
6. Brownlee, J. (05 de Mayo de 2016). *Machine Learning Mastery.* Obtenido de https://machinelearningmastery.com/introduction-python-deep-learning-library-tensorflow/
7. Canales Mora, R. (18 de Diciembre de 2020). *adictosaltrabajo.com.* Obtenido de https://www.adictosaltrabajo.com/2020/12/18/entendiendo-un-modelo-de-regresion-lineal-con-tensorboard/
8. Casas Martínez, Á. (2017). *Bibing.us.es.* Obtenido de http://bibing.us.es/proyectos/abreproy/91350/fichero/TFG+Alvaro+Casas+Martinez.pdf
9. Cubillos Figueroa, C. (Marzo de 2014). *PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO.* Obtenido de http://opac.pucv.cl/pucv_txt/txt-4500/UCE4968_01.pdf
10. Darren, T. (3 de Diciembre de 2018). *github.com.* Obtenido de https://github.com/tzutalin/labelImg
11. Del Pino, J. (11 de Abril de 2019). *MDN Web Docs.* Obtenido de https://developer.mozilla.org/es/docs/Learn/Server-side/Primeros_pasos/Web_frameworks
12. Ellingwood, J., & Kathleen, J. (5 de Diciembre de 2019). *digitalocen.com.* Obtenido de https://www.digitalocean.com/community/tutorials/como-preparar-aplicaciones-de-flask-con-gunicorn-y-nginx-en-ubuntu-18-04-es
13. García, P. P. (2013). *Universidad Complutense de Madrid.* Obtenido de https://eprints.ucm.es/23444/1/ProyectoFinMasterPedroPablo.pdf
14. Gil Rodríguez, J. L. (2008). *CENATAV.* Obtenido de Centro de Aplicaciones de Tecnologías de Avanzada: http://www.cenatav.co.cu/doc/RTecnicos/RT%20SerieAzul_004web.pdf

15. Gutiérrez, C., Fernández Medina, E., & Piattini, E. (Enero de 2005). *info-ab.uclm.es.* Obtenido de Seguridad en Servicios Web: http://www.info-ab.uclm.es/descargas/technicalreports/DIAB-05-01-2/Seguridad_en_Servicios_Web.pdf
16. *halweb.uc3m.es.* (2018). Obtenido de http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf
17. Hernández Ávila, R. (Marzo de 2018). *Research Gate.* Obtenido de Deep Fearning, una revisión: https://www.researchgate.net/publication/323858502_Deep_Learning_Una_revision
18. Kourí, G. (2006). *Instituto de Medicina Tropical.* Obtenido de http://iris.paho.org/xmlui/bitstream/handle/123456789/7992/30314.pdf;sequence=1
19. Llamas Martínez, J. (2018). *http://oa.upm.es/.* Obtenido de Universidad Politécnica de Madrid: http://oa.upm.es/53050/1/TFG_JAVIER_MARTINEZ_LLAMAS.pdf
20. Loncomilla, P. (2016). *INAOEP.* Obtenido de https://ccc.inaoep.mx/~pgomez/deep/presentations/2016Loncomilla.pdf
21. Majeda. (s.f.). *Junta de Andalucía.* Obtenido de Conceptos de seguridad en los servicios WEB: http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/211
22. Martínez, U., Ferreira, H., Espinosa, F., Hernández, A., Pintor, A., & Ramos, J. (2021). Diseño e Implementación de un Sistema Web REST API para la Detección de Mosquitos Vectores del Dengue, Basado en Deep Learning. *Memorias del Congreso Internacional, ISSN online 1946-5351*, 1012-1021.
23. Navarro-Pelayo Láinez, M. d. (30 de Agosto de 2018). *Clinic Cloud.* Obtenido de https://clinic-cloud.com/blog/protocolos-de-seguridad-de-la-informacion/
24. Oliva Rodríguez, A. (2018). *Universidad Politécnica de Valéncia.* Obtenido de https://riunet.upv.es/bitstream/handle/10251/107243/OLIVA%20-%20Desarrollo%20de%20una%20aplicaci%C3%B3n%20de%20reconocimiento%20en%20im%C3%A1genes%20utilizando%20Deep%20Learning%20con%20O....pdf?sequence=1&isAllowed=y
25. Paulus, D., & Hornegger, J. (1998). *Wikiwand.* Obtenido de Applied Pattern Recognition: https://www.wikiwand.com/es/Reconocimiento_de_patrones
26. Pérez López, O. (12 de Septiembre de 2014). *Blogspot.com.* Obtenido de Procesasamiento de Imágenes Digitales: http://oscarprzlopz.blogspot.com/2014/09/procesasamiento-de-imagenes-digitales.html
27. Ramírez Q, J., & Chacón M, M. (2011). Redes neuronales artificiales para el procesamiento. *RIEE&C, Revista de Ingeniería Eléctrica, Electrónica y Computación*, 7-16.
28. Rodríguez Araújo, J. (2018). *Planeta Chatbot.* Obtenido de https://planetachatbot.com/c%C3%B3mo-empezar-con-reconocimiento-de-imagen-y-redes-neuronales-convolucionales-en-5-minutos-7f651054dfd7

29. Sanchez Ortiz, A., Fierro Radilla, A., Nakano Miyatake, M., & Robles Camarillo, D. (Enero de 2017). *www.researchgate.net.* Obtenido de https://www.researchgate.net/publication/315852040_Mosquito_larva_classification_method_based_on_convolutional_neural_networks
30. Tébar, E. (2020). *We are marketing.* Obtenido de https://www.wearemarketing.com/es/blog/frameworks-en-el-desarrollo-web-las-mejores-practicas-para-tu-negocio-online.html
31. *Universidad Internacional de Valencia.* (19 de Febrero de 2019). Obtenido de https://www.universidadviu.com/reconocimiento-de-imagenes-software-y-ejemplos/
32. Vázquez Rull, M. (2016). *Bibing.us.es.* Obtenido de http://bibing.us.es/proyectos/abreproy/91070/fichero/Marina+Vazquez+Rull+-+Reconocimiento+de+Objetos+usando+Deep+Learning+TFG.pdf

APÉNDICE A. Instalación y configuración del Framework

Instalación de Anaconda y TensorFlow

```
bash ~/Descargas/Anaconda3-2020.07-Linux-
x86_64.sh

conda create --name entrenamientomosquito

conda install python=3.7 anaconda=custom

pip install tensorflow=1.15
```

Configuración TensorFlow

```
# Faster R-CNN with Resnet-101 (v1)
configuration for MSCOCO Dataset.
# Users should configure the
fine_tune_checkpoint field in the train
config as
# well as the label_map_path and
input path fields in the
train_input_reader and
# eval_input_reader. Search for
"PATH_TO_BE_CONFIGURED" to find the fields
that
# should be configured.

model {
  faster_rcnn {
    num_classes: 2
    image_resizer {
      keep_aspect_ratio_resizer {
        min dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_resnet101'
      first_stage_features_stride: 16
    }
    first stage anchor generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }

first_stage_box_predictor_conv_hyperparams
{
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
    }
    first stage nms score threshold: 0.0
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
    first_stage_localization_loss_weight:
2.0
    first_stage_objectness_loss_weight:
1.0
    initial_crop_size: 14
    maxpool kernel size: 2
    maxpool_stride: 2
    second_stage_box_predictor {
      mask_rcnn_box_predictor {
        use dropout: false
        dropout_keep_probability: 1.0
        fc hyperparams {
          op: FC
          regularizer {
            l2_regularizer {
              weight: 0.0
            }
          }
          initializer {
            variance_scaling_initializer {
              factor: 1.0
              uniform: true
              mode: FAN_AVG
            }
          }
        }
      }
    }
    second stage post processing {
      batch_non_max_suppression {
        score_threshold: 0.0
        iou_threshold: 0.6
        max_detections_per_class: 100
        max_total_detections: 300
      }
      score_converter: SOFTMAX
    }
    second_stage_localization_loss_weight:
2.0

second_stage_classification_loss_weight:
1.0
  }
}

train_config: {
  batch_size: 1
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial learning rate: 0.0003
          schedule {
            step: 0
            learning_rate: .0003
          }
          schedule {
            step: 900000
            learning_rate: .00003
          }
          schedule {
            step: 1200000
            learning_rate: .000003
          }
        }
```

```
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  gradient_clipping_by_norm: 10.0
  fine_tune_checkpoint:
"modelo/model.ckpt"
  from_detection_checkpoint: true
  # Note: The below line limits the
training process to 200K steps, which we
  # empirically found to be sufficient
enough to train the pets dataset. This
  # effectively bypasses the learning rate
schedule (the learning rate will
  # never decay). Remove the below line to
train indefinitely.
  num_steps: 200000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}

train_input_reader: {
  tf_record_input_reader {
    input_path:
"TFRecords/entrenamiento.record"
  }
  label map path:
"configuracion/label_map.pbtxt"
}

eval config: {
  num examples: 8000
  # Note: The below line limits the
evaluation process to 10 evaluations.
  # Remove the below line to evaluate
indefinitely.
  max_evals: 10
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "TFRecords/test.record"
  }
  label map path:
"configuracion/label_map.pbtxt"
  shuffle: false
  num_readers: 1
  num_epochs: 1
}
```

```
item {
  id: 1
  name: 'Aedes aegypti'
}

item {
  id: 2
  name: 'Aedes albopictus'
}
```

```
export
PYTHONPATH=$PYTHONPATH:'pwd':'pwd'/slim

python xml_a_csv.py --inputs=img_test --
output=test
python xml a csv.py --
inputs=img_entrenamiento --
output=entrenamiento
python csv_a_tf.py --
csv_input=CSV/test.csv --
output_path=TFRecords/test.record --
images=images
python csv_a_tf.py --
csv_input=CSV/entrenamiento.csv --
output_path=TFRecords/entrenamiento.record
--images=images

pip install -e .
protoc object_detection/protos/*.proto --
python_out=.
export PYTHONPATH=$PYTHONPATH:pwd:pwd/slim

python object detection/train.py --
logtostderr --train_dir=train --
pipeline_config_path=modelo/faster_rcnn_re
snet101_coco.config
python
object_detection/export_inference_graph.py
--input type image tensor --
pipeline_config_path
modelo/faster rcnn resnet101_coco.config
--trained_checkpoint_prefix
train/model.ckpt-684 --output_directory
modelo_congelado

python
object_detection/object_detection_runner.p
y
```

APÉNDICE B. Código del Web Service y la CNN con TensorFlow

Código del Web Service con Django

```
from django.db import models

class Imagen(models.Model):
        folio           =
models.CharField(max_length=50,
null=False)
        url_image               =
models.CharField(max_length=2082,
null=False)
        similitud       =
models.CharField(max_length=10, null=True)

        def __str__(self):
                return self.folio
```

```
import os
import pandas as pd
from collections import namedtuple,
OrderedDict
from django.shortcuts import render
from .models import Imagen
from api import serializers
from rest_framework.views import APIView
from rest_framework.response import
Response
from rest_framework import status
from rest_framework.authtoken.models
import Token
from django.contrib.auth.models import
User
from rest_framework.permissions import
IsAuthenticated

class ImagenTokenView(APIView):
    def get(self, request):
        user =
User.objects.get(username='adminapi')
        token =
Token.objects.create(user=user)

        return Response(
                    {
                        'messaje': 'Token
generado'
                    }
                )

class ImagenListView(APIView):
    serializer_class =
serializers.ImagenSerializer
    permission_classes = [IsAuthenticated]

    def get(self, request):
        imagenList = Imagen.objects.all()
        return
Response(list(imagenList.values()))

    def post(self, request):
        serializer =
self.serializer_class(data=request.data)
        if serializer.is_valid():
            folioImgAnte =
serializer.validated_data.get('folio')
            detailImg =
Imagen.objects.filter(folio=folioImgAnte).
first()
            if detailImg:
                return Response(
                        {
                            'Mensaje':
'Folio utilizado anteriomente, inserte un
folio nuevo',
                        }
                    )
            else:
                url_image =
serializer.validated_data.get('url_image')
                os.system('python
/home/umartinez/mosquito analisis/detectar
_auto/object_detection/object_detection_ru
nner_single.py --images=' + url_image)
                examples =
pd.read_csv('/home/umartinez/mosquito_anal
isis/detection_results/prueva_csv.csv')
                values_folio =
examples['Folio'].unique()
                values_similitud =
examples['Similitud'].unique()
                if values_folio[0] ==
'Error':
                    return Response(
                        {
                            'Mensaje':
'Error al analizar imagen',
                            'Exception ':
values_similitud[0],
                        }
                    )

                serializer.save()
                for index_folio in
range(0, len(examples)):
                    folio_save =
values_folio[index_folio]
                    similitud_save =
values_similitud[index_folio]
                    folioImg =
serializer.validated_data.get('folio')
                    detailImg =
Imagen.objects.filter(folio=folioImg).firs
t()
                    detailImg.similitud =
similitud_save
                    detailImg.save()

                similitudImg =
detailImg.similitud
                statusDetec = False
```

```
                arraySimi =
detailImg.similitud.split(':')
                arraySimilitud =
arraySimi[1].split(' ')
                arraySimiValAnt =
arraySimi[1].split('%')
                arraySimiVal =
arraySimiValAnt[0].split(' ')
                arraySimiValTipo =
arraySimi[0]

                if int(arraySimiVal[1]) >
70:
                    statusDetec = True
                return Response(
                    {
                        'Folio': folioImg,
                        'Status':
statusDetec,
                        'Tipo':
arraySimiValTipo,
                        'Similitud':
arraySimilaridadSimilitud[1]
                    }
                )
        else:
            return Response(
                serializer.errors,

status=status.HTTP_400_BAD_REQUEST
            )

class ImagenDetailView(APIView):
    permission_classes = [IsAuthenticated]

    def get(self, request, folio):
        detailImg =
Imagen.objects.filter(folio=folio).first()
        if detailImg:
            statusDetec = False
            arraySimi =
detailImg.similitud.split(':')
            arraySimilitud =
arraySimi[1].split(' ')
            arraySimiValAnt =
arraySimi[1].split('%')
            arraySimiVal =
arraySimiValAnt[0].split(' ')
            arraySimiValTipo =
arraySimi[0].split(':')
            if int(arraySimiVal[1]) > 70:
                statusDetec = True
            return Response(
                    {
                        'Folio':
detailImg.folio,
                        'Status':
statusDetec,
                        'Tipo':
arraySimiValTipo[0],
                        'Similitud':
arraySimilaridadSimilitud[1],
                        'Url_Image':
detailImg.url_image
                    }
                )
        else:
            return Response(
                {
                    'Mensaje': 'Folio no
encontrado',
                }
            )
```

Código de Python y TensorFlow

```
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET
import argparse, sys

parser = argparse.ArgumentParser()
parser.add_argument('--inputs',
help='Directorio con los XMLs')
parser.add_argument('--output',
help='Nombre del CSV a generarse')
args = parser.parse_args()
def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path +
'/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in
root.findall('object'):
            value =
(root.find('filename').text,

int(root.find('size')[0].text),

int(root.find('size')[1].text),
                     member[0].text,

int(member[4][0].text),

int(member[4][1].text),

int(member[4][2].text),

int(member[4][3].text)
                     )
            xml_list.append(value)
    column_name = ['filename', 'width',
'height', 'class', 'xmin', 'ymin', 'xmax',
'ymax']
    xml_df = pd.DataFrame(xml_list,
columns=column_name)
    return xml_df

def main():
    image_path = os.path.join(os.getcwd(),
args.inputs)
    xml_df = xml_to_csv(image_path)

xml_df.to_csv(os.path.join(os.getcwd(),'CS
V','{}.csv'.format(args.output)),
index=None)
    print('Se han generado los CSVs')

main()
```

```
from __future__ import division
from __future__ import print_function
from __future__ import absolute_import
from object_detection.utils import
label_map_util
import os
import io
import pandas as pd
import tensorflow as tf

from PIL import Image
from object_detection.utils import
dataset_util
from collections import namedtuple,
OrderedDict

flags = tf.app.flags
flags.DEFINE_string('csv_input', '', 'Path
to the CSV input')
flags.DEFINE_string('output_path', '',
'Path to output TFRecord')
flags.DEFINE_string('images', '', 'Path to
the images')
FLAGS = flags.FLAGS

def class_text_to_int(row_label):
    label_map_dict =
label_map_util.get_label_map_dict('configu
racion/label_map.pbtxt')
    identifier = label_map_dict[row_label]
    print(label_map_dict)
    return identifier

def split(df, group):
    data = namedtuple('data', ['filename',
'object'])
    gb = df.groupby(group)
    return [data(filename,
gb.get_group(x)) for filename, x in
zip(gb.groups.keys(), gb.groups)]

def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path,
'{}'.format(group.filename)), 'rb') as
fid:
        encoded_jpg = fid.read()
    encoded_jpg_io =
io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size

    filename =
group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
    classes = []

    for index, row in
group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)

classes_text.append(row['class'].encode('u
tf8'))

classes.append(class_text_to_int(row['clas
s']))
```

```
    tf_example =
tf.train.Example(features=tf.train.Feature
s(feature={
        'image/height':
dataset_util.int64_feature(height),
        'image/width':
dataset_util.int64_feature(width),
        'image/filename':
dataset_util.bytes_feature(filename),
        'image/source_id':
dataset_util.bytes_feature(filename),
        'image/encoded':
dataset_util.bytes_feature(encoded_jpg),
        'image/format':
dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin':
dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax':
dataset_util.float_list_feature(xmaxs),
        'image/object/bbox/ymin':
dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax':
dataset_util.float_list_feature(ymaxs),
        'image/object/class/text':
dataset_util.bytes_list_feature(classes_te
xt),
        'image/object/class/label':
dataset_util.int64_list_feature(classes),
    }))
    return tf_example

def main(_):
    writer =
tf.python_io.TFRecordWriter(FLAGS.output_p
ath)
    path = os.path.join(os.getcwd(),
FLAGS.images)
    examples =
pd.read_csv(FLAGS.csv_input)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example =
create_tf_example(group, path)

writer.write(tf_example.SerializeToString(
))

    writer.close()
    output_path =
os.path.join(os.getcwd(),
FLAGS.output_path)
    print('Successfully created the
TFRecords: {}'.format(output_path))

if __name__ == '__main__':
    tf.app.run()
```

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile
import json
import time
import glob
from io import StringIO
from PIL import Image
import matplotlib.pyplot as plt
from utils import visualization_utils as
vis_util
from utils import label_map_util
from multiprocessing.dummy import Pool as
ThreadPool
import argparse, sys
import pandas as pd
import xml.etree.ElementTree as ET

parser = argparse.ArgumentParser()
parser.add_argument('--labels',
help='Directorio a label_map.pbtxt',
default
='/home/umartinez/mosquito_analisis/detect
ar_auto/configuracion/label_map.pbtxt')
parser.add_argument('--images',
help='Directorio de las imagenes a
procesar', default =
'/home/umartinez/mosquito_analisis/repo_im
ages')
parser.add_argument('--model',
help='Directorio al modelo congelado',
default =
'/home/umartinez/mosquito_analisis/detecta
r_auto/modelo_congelado')
args = parser.parse_args()

MAX_NUMBER_OF_BOXES = 30
MINIMUM_CONFIDENCE = 0.4

PATH_TO_LABELS = args.labels
PATH_TO_TEST_IMAGES_DIR = args.images

label_map =
label_map_util.load_labelmap(PATH_TO_LABEL
S)
categories =
label_map_util.convert_label_map_to_catego
ries(label_map,
max_num_classes=sys.maxsize,
use_display_name=True)
CATEGORY_INDEX =
label_map_util.create_category_index(categ
ories)

# Path to frozen detection graph. This is
the actual model that is used for the
object detection.
MODEL_NAME = args.model
PATH_TO_CKPT = MODEL_NAME +
'/frozen_inference_graph.pb'

def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return
np.array(image.getdata()).reshape(
        (im_height, im_width,
3)).astype(np.uint8)

def detect_objects(image_path):
    try:
        image = Image.open(image_path)
    except Exception as e:
```

```
        print("Ha ocurrido un error =>",
type(e).__name__)
        return 'Error---Imagen no
encontrada'

    image_np =
load_image_into_numpy_array(image)
    image_np_expanded =
np.expand_dims(image_np, axis=0)

    (boxes, scores, classes, num) =
sess.run([detection_boxes,
detection_scores, detection_classes,
num_detections], feed_dict={image_tensor:
image_np_expanded})

    try:
        porcet_similitud =
vis_util.visualize_boxes_and_labels_on_ima
ge_array(
            image_np,
            np.squeeze(boxes),

np.squeeze(classes).astype(np.int32),
            np.squeeze(scores),
            CATEGORY_INDEX,

min_score_thresh=MINIMUM_CONFIDENCE,

use_normalized_coordinates=True,
            line_thickness=8)
    except Exception as e:
        print("Ha ocurrido un error =>",
type(e).__name__)
        return 'Error---No se detectaron
mosquitos en la imagen'

    fig = plt.figure()
    fig.set_size_inches(16, 9)
    ax = plt.Axes(fig, [0., 0., 1., 1.])
    ax.set_axis_off()
    fig.add_axes(ax)

    array_image_path =
image_path.split("/")
    name_image =
array_image_path[len(array_image_path) -
1]

    plt.imshow(image_np, aspect = 'auto')

plt.savefig('/home/umartinez/mosquito_anal
isis/analyzed_images/{}'.format(name_image
), dpi = 62)
    plt.close(fig)
    info = name_image + '---' +
porcet similitud
    print("info.....")
    print(info)
    return info

img_to_detect = PATH_TO_TEST_IMAGES_DIR
print('img_to_detect ', img_to_detect)

# Load model into memory
print('Loading model...')
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.compat.v1.GraphDef()
    with tf.io.gfile.GFile(PATH_TO_CKPT,
'rb') as fid:
        serialized_graph = fid.read()

od_graph_def.ParseFromString(serialized_gr
aph)
        tf.import_graph_def(od_graph_def,
name='')

print('detecting...')
resultList = []
with detection_graph.as_default():
    with
tf.compat.v1.Session(graph=detection_graph
) as sess:
        image_tensor =
detection_graph.get_tensor_by_name('image_
tensor:0')
        detection_boxes =
detection_graph.get_tensor_by_name('detect
ion_boxes:0')
        detection_scores =
detection_graph.get_tensor_by_name('detect
ion_scores:0')
        detection_classes =
detection_graph.get_tensor_by_name('detect
ion_classes:0')
        num_detections =
detection_graph.get_tensor_by_name('num_de
tections:0')

        result =
detect_objects(img_to_detect)
        resultList.append(result)

print('resultList...')
print(resultList)

def xml_to_csv(listResult):
    xml_list = []
    for member in listResult:
            values_list = member.split("--
-")
            value = (values_list[0],
                     values_list[1],
                     )
            xml_list.append(value)

    column_name = ['Folio', 'Similitud']
    xml_df = pd.DataFrame(xml_list,
columns=column_name)
    return xml_df


def main():
    image_path = resultList
    xml_df = xml_to_csv(image_path)

xml_df.to_csv(os.path.join(os.getcwd(),'/h
ome/umartinez/mosquito_analisis/detection_
results','{}.csv'.format('prueva_csv')),
index=None)
    print('Se han generado los CSVs')

main()
```

```
# Copyright 2017 The TensorFlow Authors.
All Rights Reserved.
#
# Licensed under the Apache License,
Version 2.0 (the "License");
# you may not use this file except in
compliance with the License.
# You may obtain a copy of the License at
#
#
http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or
agreed to in writing, software
# distributed under the License is
distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied.
# See the License for the specific
language governing permissions and
# limitations under the License.
#
==========================================
==================================

import collections
import numpy as np
import PIL.Image as Image
import PIL.ImageColor as ImageColor
import PIL.ImageDraw as ImageDraw
import PIL.ImageFont as ImageFont
import six
import tensorflow as tf


_TITLE_LEFT_MARGIN = 10
_TITLE_TOP_MARGIN = 10
STANDARD_COLORS = [
    'AliceBlue', 'Chartreuse', 'Aqua',
'Aquamarine', 'Azure', 'Beige', 'Bisque',
    'BlanchedAlmond', 'BlueViolet',
'BurlyWood', 'CadetBlue', 'AntiqueWhite',
    'Chocolate', 'Coral',
'CornflowerBlue', 'Cornsilk', 'Crimson',
'Cyan',
    'DarkCyan', 'DarkGoldenRod',
'DarkGrey', 'DarkKhaki', 'DarkOrange',
    'DarkOrchid', 'DarkSalmon',
'DarkSeaGreen', 'DarkTurquoise',
'DarkViolet',
    'DeepPink', 'DeepSkyBlue',
'DodgerBlue', 'FireBrick', 'FloralWhite',
    'ForestGreen', 'Fuchsia', 'Gainsboro',
'GhostWhite', 'Gold', 'GoldenRod',
    'Salmon', 'Tan', 'HoneyDew',
'HotPink', 'IndianRed', 'Ivory', 'Khaki',
    'Lavender', 'LavenderBlush',
'LawnGreen', 'LemonChiffon', 'LightBlue',
    'LightCoral', 'LightCyan',
'LightGoldenRodYellow', 'LightGray',
'LightGrey',
    'LightGreen', 'LightPink',
'LightSalmon', 'LightSeaGreen',
'LightSkyBlue',
    'LightSlateGray', 'LightSlateGrey',
'LightSteelBlue', 'LightYellow', 'Lime',
    'LimeGreen', 'Linen', 'Magenta',
'MediumAquaMarine', 'MediumOrchid',
    'MediumPurple', 'MediumSeaGreen',
'MediumSlateBlue', 'MediumSpringGreen',
    'MediumTurquoise', 'MediumVioletRed',
'MintCream', 'MistyRose', 'Moccasin',
    'NavajoWhite', 'OldLace', 'Olive',
'OliveDrab', 'Orange', 'OrangeRed',
    'Orchid', 'PaleGoldenRod',
'PaleGreen', 'PaleTurquoise',
'PaleVioletRed',
    'PapayaWhip', 'PeachPuff', 'Peru',
'Pink', 'Plum', 'PowderBlue', 'Purple',
    'Red', 'RosyBrown', 'RoyalBlue',
'SaddleBrown', 'Green', 'SandyBrown',
    'SeaGreen', 'SeaShell', 'Sienna',
'Silver', 'SkyBlue', 'SlateBlue',
    'SlateGray', 'SlateGrey', 'Snow',
'SpringGreen', 'SteelBlue', 'GreenYellow',
    'Teal', 'Thistle', 'Tomato',
'Turquoise', 'Violet', 'Wheat', 'White',
    'WhiteSmoke', 'Yellow', 'YellowGreen'
]


def save_image_array_as_png(image,
output_path):
  image_pil =
Image.fromarray(np.uint8(image)).convert('
RGB')
  with tf.gfile.Open(output_path, 'w') as
fid:
    image_pil.save(fid, 'PNG')


def encode_image_array_as_png_str(image):

  image_pil =
Image.fromarray(np.uint8(image))
  output = six.BytesIO()
  image_pil.save(output, format='PNG')
  png_string = output.getvalue()
  output.close()
  return png_string


def
draw_bounding_box_on_image_array(image,
                                 ymin,
                                 xmin,
                                 ymax,
                                 xmax,

color='red',

thickness=4,

display_str_list=(),

use_normalized_coordinates=True):

  image_pil =
Image.fromarray(np.uint8(image)).convert('
RGB')
  draw_bounding_box_on_image(image_pil,
ymin, xmin, ymax, xmax, color,
                             thickness,
display_str_list,
```

```
use_normalized_coordinates)
  np.copyto(image, np.array(image_pil))


def draw_bounding_box_on_image(image,
                               ymin,
                               xmin,
                               ymax,
                               xmax,

color='red',

thickness=4,

display_str_list=(),

use_normalized_coordinates=True):

  draw = ImageDraw.Draw(image)
  im_width, im_height = image.size
  if use_normalized_coordinates:
    (left, right, top, bottom) = (xmin *
im_width, xmax * im_width,
                                  ymin *
im_height, ymax * im_height)
  else:
    (left, right, top, bottom) = (xmin,
xmax, ymin, ymax)
  draw.line([(left, top), (left, bottom),
(right, bottom),
             (right, top), (left, top)],
width=thickness, fill=color)
  try:
    font = ImageFont.truetype('arial.ttf',
24)
  except IOError:
    font = ImageFont.load_default()

  text_bottom = top
  # Reverse list and print from bottom to
top.
  for display_str in display_str_list[::-
1]:
    text_width, text_height =
font.getsize(display_str)
    margin = np.ceil(0.05 * text_height)
    draw.rectangle(
        [(left, text_bottom - text_height
- 2 * margin), (left + text_width,

text_bottom)],
        fill=color)
    draw.text(
        (left + margin, text_bottom -
text_height - margin),
        display_str,
        fill='black',
        font=font)
    text_bottom -= text_height - 2 *
margin


def
draw_bounding_boxes_on_image_array(image,

boxes,

color='red',

thickness=4,

display_str_list_list=()):

  image_pil = Image.fromarray(image)
  draw_bounding_boxes_on_image(image_pil,
boxes, color, thickness,

display_str_list_list)
  np.copyto(image, np.array(image_pil))


def draw_bounding_boxes_on_image(image,
                                 boxes,

color='red',

thickness=4,

display_str_list_list=()):

  boxes_shape = boxes.shape
  if not boxes_shape:
    return
  if len(boxes_shape) != 2 or
boxes_shape[1] != 4:
    raise ValueError('Input must be of
size [N, 4]')
  for i in range(boxes_shape[0]):
    display_str_list = ()
    if display_str_list_list:
      display_str_list =
display_str_list_list[i]
    draw_bounding_box_on_image(image,
boxes[i, 0], boxes[i, 1], boxes[i, 2],
                               boxes[i,
3], color, thickness, display_str_list)


def draw_keypoints_on_image_array(image,

keypoints,

color='red',

radius=2,

use_normalized_coordinates=True):

  image_pil =
Image.fromarray(np.uint8(image)).convert('
RGB')
  draw_keypoints_on_image(image_pil,
keypoints, color, radius,

use_normalized_coordinates)
  np.copyto(image, np.array(image_pil))


def draw_keypoints_on_image(image,
                            keypoints,
                            color='red',
                            radius=2,

use_normalized_coordinates=True):
```

```
draw = ImageDraw.Draw(image)
  im_width, im_height = image.size
  keypoints_x = [k[1] for k in keypoints]
  keypoints_y = [k[0] for k in keypoints]
  if use_normalized_coordinates:
    keypoints_x = tuple([im_width * x for
x in keypoints_x])
    keypoints_y = tuple([im_height * y for
y in keypoints_y])
  for keypoint_x, keypoint_y in
zip(keypoints_x, keypoints_y):
    draw.ellipse([(keypoint_x - radius,
keypoint_y - radius),
                  (keypoint_x + radius,
keypoint_y + radius)],
                 outline=color,
fill=color)


def draw_mask_on_image_array(image, mask,
color='red', alpha=0.7):

  if image.dtype != np.uint8:
    raise ValueError('`image` not of type
np.uint8')
  if mask.dtype != np.float32:
    raise ValueError('`mask` not of type
np.float32')
  if np.any(np.logical_or(mask > 1.0, mask
< 0.0)):
    raise ValueError('`mask` elements
should be in [0, 1]')
  rgb = ImageColor.getrgb(color)
  pil_image = Image.fromarray(image)

  solid_color = np.expand_dims(
      np.ones_like(mask), axis=2) *
np.reshape(list(rgb), [1, 1, 3])
  pil_solid_color =
Image.fromarray(np.uint8(solid_color)).con
vert('RGBA')
  pil_mask =
Image.fromarray(np.uint8(255.0*alpha*mask)
).convert('L')
  pil_image =
Image.composite(pil_solid_color,
pil_image, pil_mask)
  np.copyto(image,
np.array(pil_image.convert('RGB')))


def
visualize_boxes_and_labels_on_image_array(
image,
boxes,
classes,
scores,
category_index,
instance_masks=None,
keypoints=None,
use_normalized_coordinates=False,
max_boxes_to_draw=20,
min_score_thresh=.5,
agnostic_mode=False,
line_thickness=4,
result=""):
box_to_display_str_map =
collections.defaultdict(list)
  box_to_color_map =
collections.defaultdict(str)
  box_to_instance_masks_map = {}
  box_to_keypoints_map =
collections.defaultdict(list)
  if not max_boxes_to_draw:
    max_boxes_to_draw = boxes.shape[0]
  more_simi = 0
  display_str_final = ''
  for i in range(min(max_boxes_to_draw,
boxes.shape[0])):
    if scores is None or scores[i] >
min_score_thresh:
      box = tuple(boxes[i].tolist())
      if instance_masks is not None:
        box_to_instance_masks_map[box] =
instance_masks[i]
      if keypoints is not None:

box_to_keypoints_map[box].extend(keypoints
[i])
      if scores is None:
        box_to_color_map[box] = 'black'
      else:
        if not agnostic_mode:
          if classes[i] in
category_index.keys():
            class_name =
category_index[classes[i]]['name']
          else:
            class_name = 'N/A'
          if int(100*scores[i]) >
more_simi:
            display_str_final = '{}:
{}%'.format(
                class_name,
                int(100*scores[i]))
            more_simi = int(100*scores[i])
          display_str = '{}: {}%'.format(
              class_name,
              int(100*scores[i]))
        else:
          display_str = 'score:
{}%'.format(int(100 * scores[i]))

box_to_display_str_map[box].append(display
_str)
        if agnostic_mode:
          box_to_color_map[box] =
'DarkOrange'
        else:
          box_to_color_map[box] =
STANDARD_COLORS[
              classes[i] %
len(STANDARD_COLORS)]

  # Draw all boxes onto image.
  for box, color in
box_to_color_map.items():
    ymin, xmin, ymax, xmax = box
    if instance_masks is not None:
      draw_mask_on_image_array(
          image,
          box_to_instance_masks_map[box],
          color=color
```

```
      )
    draw_bounding_box_on_image_array(
        image,
        ymin,
        xmin,
        ymax,
        xmax,
        color=color,
        thickness=line_thickness,

display_str_list=box_to_display_str_map[bo
x],

use_normalized_coordinates=use_normalized_
coordinates)
    if keypoints is not None:
      draw_keypoints_on_image_array(
          image,
          box_to_keypoints_map[box],
          color=color,
          radius=line_thickness / 2,

use_normalized_coordinates=use_normalized_
coordinates)
  return display_str_final
```

APÉNDICE C. Carátula del Artículo Academia Journals

Memorias del Congreso Internacional
de Investigación Academia Journals
Morelia 2021

Morelia, Michoacán, México
12 al 14 de mayo, 2021

Diseño e Implementación de un Sistema Web REST API para la Detección de Mosquitos Vectores del Dengue, Basado en Deep Learning

Ulises Martínez Covarrubias[1], Heberto Ferreira Medina[2], Francisco Javier Espinosa García[3], Anastacio Antolino Hernández[4], Abel Pintor Estrada[5], José Guadalupe Ramos Díaz[6]

Resumen— En este trabajo se analizan los patrones en imágenes de mosquito adquiridas desde una aplicación móvil, para detectar la presencia de mosquitos *Aedes aegypti* y *Aedes albopictus* en poblaciones urbanas, que generan un problema de salud pues son vectores de la enfermedad del Dengue, Zika y Chikungunya. Se desarrolla un sistema de software REST API basado en una aplicación Web y el lenguaje Python que implementa redes neuronales convolutivas (CNN) para el Deep Learning (DL) mediante el uso de la biblioteca TensorFlow y Keras. Se utiliza un servidor Web seguro con el framework Django, el cual implementa la comunicación entre la aplicación móvil y el sistema de DL. Las funciones del servicio Web son: 1) publicar el servicio para enviar imágenes, 2) almacenarlas y clasificarlas y 3) analizar la similitud con los mosquitos. Los resultados son guardados en una base datos para su posterior consulta a través del sistema móvil o Web. Se proporciona el informe de los análisis y el porcentaje de similitud de imagen probada. La metodología utilizada en este proyecto sigue las fases; 1) almacenamiento de imágenes, 2) tratamiento utilizando DL, 3) uso de CNN, 4) uso del modelo de aprendizaje para establecer la similitud y 5) reporte de resultados. Los resultados del sistema permiten garantizar que la tecnología desarrollada es confiable en la detección de mosquitos *Aedes aegypti* y *Aedes albopictus*, por lo que su potencial industrial es viable.

Palabras clave— Deep Learning, Dengue, CNN, TensorFlow, modelo de aprendizaje.

Introducción

El dengue es una enfermedad desarrollada por un virus y su vector de transmisión es por diferentes especies de mosquitos que se ha propagado rápidamente en muchas regiones del planeta y que la Organización Mundial de la Salud (OMS) reconoce como un problema de salud muy importante en los últimos años. El virus del dengue se transmite por mosquitos hembra de *Aedes (Stegomya) aegypti* y por la de *Aedes (Stegomya) albopictus*. Estos mosquitos, comúnmente llamados "mosquitos tigre" también transmiten la fiebre Chikunguya, la fiebre amarilla y el virus del Zika. Estas enfermedades están muy extendidas en los países tropicales, con variaciones locales que dependen en gran medida de las precipitaciones, la temperatura y la urbanización rápida sin planificar. México ha presentado en años recientes un brote importante en los estados del centro sur del país (Bhatt, Gething, Brady, Messina, Farlow, & Moyes, 2019).

Las personas infectadas sintomáticas y asintomáticas son los portadores y multiplicadores principales del virus, y los mosquitos se infectan y convierten en vectores cuando se alimentan de esas personas. Tras la aparición de los primeros síntomas, las personas infectadas pueden transmitirla (durante 4 a 12 días como máximo) a los mosquitos

[1] Ingeniero por el Instituto Tecnológico Superior de Puruándiro y estudiante de la Maestría en Sistemas Computacionales (MSC) del Departamento de Sistemas y Computación (DSC) del Tecnológico Nacional de México (TecNM) Campus Instituto Tecnológico de Morelia (I.T. Morelia), Michoacán, México. ulises.umc@gmail.com.

[2] Doctor en Tecnologías de Información y Análisis de Decisiones por la Universidad Popular Autónoma del Estado de Puebla (UPAEP), es encargado de telecomunicaciones de la Unidad de TICS Instituto de Investigaciones Ecosistemas y Sustentabilidad (IIES), UNAM y profesor de la MSC del DSC del TecNM/I.T. Morelia, Michoacán, México. hferreir@iies.unam.mx. (***autor corresponsal***)

[3] Doctor en Biología (Ph.D.) por la Universidad de California, Santa Cruz, especializado en Ecología Química, Agroecología y en Ecología de plantas invasoras y malezas, es investigador en el Instituto de Investigaciones Ecosistemas y Sustentabilidad de la UNAM y profesor del Posgrado en Ciencias Biológicas UNAM y de la Licenciatura en Ciencias Ambientales de la Escuela Nacional Estudios Superiores Morelia, UNAM. espinosa@iies.unam.mx.

[4] Doctor en Ciencias en Ingeniería Eléctrica por el Depto. de Estudios de Posgrado de la Facultad de Ingeniería Eléctrica (DEP-FIE) de la UMSNH, es profesor de la MSC del DSC del TecNM/I.T. Morelia, Michoacán. México. anastacio.ah@morelia.tecnm.mx.

[5] Maestro en Ciencias en Ciencias de la Computación por el Departamento de Sistemas y Computación del TecNM/I.T. Morelia, es profesor de la MSC del TecNM/I.T., Morelia, Michoacán. México. abel.pe@morelia.tecnm.mx.

[6] Doctor en Informática por el DSIC de la Universidad Politécnica de Valencia y profesor del DSC del TecNM/I.T. Morelia, Michoacán, México. jose.rd@morelia.tecnm.mx