

Sistema automático para la detección de baches en la malla vial de Bogotá asistido por imágenes digitales.



**UNIVERSIDAD
CENTRAL**

Juan Esteban Puyo Cubillos
Johan Santiago Gómez Guillen
Laura Alejandra Rocha Rincones

Universidad Central
Inteligencia Artificial

Hugo Franco Triana

26 de Mayo 2021

Contenido

Introducción	5
Contextualización y definición del problema	5
Antecedentes	6
Malla vial en Bogotá	6
Trabajos previos y relacionados	7
Objetivo general	9
Propuesta metodológica del proyecto	9
Especificación del entorno del trabajo	10
Clasificación del entorno de trabajo	11
Datos	11
Materiales y Métodos	12
Resultados	17
Discusión	19
REFERENCIAS	20
Anexo A. Código y dataset del proyecto Clasificación de baches	23

Lista de figuras

Figura 1. Metodología del proyecto	11
Figura 2. Herramienta LabelImg	14
Figura 3. Funcionamiento de una red convolucional	16
Figura 4. Grilla creada por YOLO para la predicción.	17
Figura 5. Métrica IoU para la precisión de la detección de objetos.	17
Figura 6. Procedimiento que sigue el algoritmo para etiquetar objetos.	18
Figura 7. Entrenamiento de la red neuronal convolucional.	19
Figura 7. Test de pruebas con imágenes aleatorias en diferentes ángulos	20
Figura 8. Test de pruebas 2 con imágenes aleatorias en diferentes ángulos	20

Lista de tablas

Tabla 1. Especificación del entorno de trabajo	11
Tabla 2. Reporte de características de los datasets	13

Lista de Símbolos y abreviaturas

- IDU: Instituto de Desarrollo Urbano.
- IoU: Intersection over Union

Introducción

Contextualización y definición del problema

Actualmente, Bogotá padece un significativo deterioro en su malla vial. En el estudio de El tiempo (2017), se afirma que “De los 15.557 kilómetros de malla vial que tiene Bogotá, el 34 por ciento se encuentra en mal estado”. Todo esto se debe, en gran medida, a la ineficiente gestión en los procesos de identificación, registro, y seguimiento que dispone la Alcaldía y su unidad de Mantenimiento vial y el IDU, entes de control encargados de la reparación de los baches. Una causa adicional, es el creciente aumento de la malla vial, ya que se prioriza la construcción de nuevas vías que la reparación de las existentes. Por último, también se debe la falta de comunicación entre los diferentes entes encargados del control de baches y los ciudadanos, ya que se restringe la posibilidad de realizar peticiones, quejas y hacer seguimiento a los proyectos de reparación en la malla vial.

El aumento constante de los baches en la ciudad de Bogotá es una de las principales causas de accidentes, congestión vial y deterioro de los vehículos. En general, es uno de los problemas que más aqueja a todos los bogotanos sin importar el estrato o el lugar donde vivan. A diario vemos en noticias y redes sociales que centenares de ciudadanos se quejan constantemente por los miles de huecos que hay por toda Bogotá.

Con base en lo anterior, el presente proyecto tiene como objetivo diseñar un sistema automático para la detección de baches en la malla vial de Bogotá asistido por imágenes digitales, el cual, identifique, reporte y comunique al conductor, la presencia de un bache a proximidad mientras está conduciendo.

Antecedentes

Malla vial en Bogotá

La malla vial es el componente principal del sistema de transporte terrestre sobre la cual se fundamenta la movilidad de bienes y personas dentro de una entidad territorial. Según Londoño (2014), el mantenimiento de la infraestructura vial es una prioridad para las autoridades encargadas de su administración, ya que representa un patrimonio muy valioso para la economía de la región en la que se encuentra.

Una malla vial en buen estado contribuye a la articulación entre los puntos de origen y destino de los diferentes viajes que se generan, lo cual se traduce en una mayor competitividad y desarrollo del transporte e infraestructura del territorio del cual hace parte. Esto se ve reflejado en la realización de viajes más rápidos, cómodos, seguros y económicos.

Por otro lado, el deterioro de la malla vial tiene repercusiones económicas, ambientales, sociales y políticas, dentro de las cuales se pueden mencionar principalmente, desde el punto de vista de los usuarios, el aumento en los costos de operación vehicular, en tiempos de desplazamiento, inconformidad, accidentalidad, congestión y consumo de combustible.

En Bogotá la política pública en materia de malla vial, ha configurado problemas estructurales, que son evidenciados en los numerosos baches y huecos presentes por toda la ciudad.

Trabajos previos y relacionados

Existen algunas investigaciones previas en el marco de la implementación de sistemas en los temas relacionados con vías. A continuación mostraremos algunos de estos importantes antecedentes.

Barreth (2018), desarrolló un sistema electrónico detector de baches que captura la latitud y longitud de donde fue detectado y almacena esta información en una base de datos para luego mostrar los marcadores geográficos por medio de un mapa en una aplicación móvil. El prototipo del sistema simula el impacto de un vehículo al caer en un bache a través de un sensor piezo-eléctrico y luego solicita al módulo GPS las coordenadas geográficas, la información es recolectada y procesada por el Microcontrolador Arduino Uno para posteriormente enviar estos datos por comando AT al módulo WiFi y este a su vez a la base de datos. Para brindar asistencia vehicular mientras el conductor se encuentra manejando Barreth desarrolló una aplicación móvil en Android Studio que utiliza Google Maps para mostrar los marcadores de los baches encontrados y mide la distancia entre la ubicación actual del conductor y los baches, permitiendo así la reproducción de un audio alertando al conductor de la cercanía de un bache de acuerdo a su gravedad alta, media o baja.

Rodríguez y otros (2016), desarrollaron una solución tecnológica que permite a los usuarios registrar los baches presentes en la malla vial de Bogotá por medio de fotografías, ubicación geográfica y demás información, permitiendo realizar seguimiento a todas sus solicitudes. Gracias a que la solución proporciona comunicación directa con las entidades de control; el sistema prioriza y atiende las solicitudes agilizando así la reparación de los baches.

Todo esto mediante una plataforma móvil para los propietarios de teléfonos inteligentes y una plataforma web para cualquier persona con conexión a internet.

Galeano y otros (2017), desarrollaron un proyecto para la detección de baches de una calzada vial a partir de imágenes de alta resolución implementando metodologías orientadas a objetos (GEOBIA). El proyecto inicia con la captura de los datos en campo en donde se busca obtener un GSD de muy alta resolución, el cual permite la detección de objeto con alto detalle, esta información es procesada mediante la creación de un algoritmo, el cual segmenta y clasifica a partir de atributos de geométricos, temáticos, espectrales, y de lógica difusa. Estos generan como resultado una metodología que permite la detección de baches o huecos dentro de una zona vial.

Maeda y otros (2018), prepararon un conjunto de datos de daños viales a gran escala. Este conjunto de datos se compone de 9.053 imágenes de daños en la carretera capturadas con un teléfono inteligente instalado en un automóvil, con 15.435 casos de daños en la superficie de la carretera incluidos en estas imágenes de la carretera. Para generar este conjunto de datos, cooperaron con 7 municipios de Japón y adquirieron imágenes de carreteras durante más de 40 horas. Estas imágenes fueron capturadas en una amplia variedad de condiciones climáticas y de iluminación. En cada imagen, anotaron el cuadro delimitador que representa la ubicación y el tipo de daño.

Por último, Manzanares (2019), desarrolló un detector de baches con tecnología Deep Learning. Su principal aplicación es reducir el coste del mantenimiento de las carreteras, automatizando y agilizando el trabajo, y además dar un uso particular, donde los usuarios de la carretera puedan mejorar su seguridad mediante el aviso del bache o adaptando la suspensión

electrónica del vehículo para poder absorberlo mejor. Para cumplir con el objetivo utilizó Deep Learning, ya que, según él es una las tecnologías más actuales y con mejores resultados en el campo de la detección de objetos y está en expansión y mejora continua. En segundo lugar creó un detector de baches a partir de esta tecnología y por último se aplicó todo esto a una APP como un prototipo funcional de detector de baches.

Objetivo general

Diseñar un prototipo funcional de un sistema para la detección de baches en la malla vial de Bogotá empleando tecnologías de visión por computador supervisado y técnicas propias del aprendizaje profundo como el entrenamiento y clasificación de imágenes ,el cual, capture imágenes, compare con el dataset dado, detecte a distancia y notifique al conductor la presencia de un bache mientras está conduciendo.

Propuesta metodológica del proyecto

Para el desarrollo del proyecto se hizo uso de la siguiente metodología definida en seis fases:

- Análisis y definición del problema: En esta se identificó la problemática central, sus causas, consecuencias, factores y actores involucrados
- Especificación y clasificación del entorno de trabajo: Esta contiene el problema que debe resolver el agente a nivel de su diseño e implementación para convertirlo en una solución.
- Propuesta de solución: En esta se define la metodología y algoritmos a utilizar para la identificación de diferentes tipos de baches.

- Especificación de herramientas: En esta se especificará las herramientas, librerías y los datos necesarios para la implementación del proyecto.
- Implementación: A partir del entorno, algoritmos, herramientas y librerías especificadas se procederá hacer uso de los mismo para llegar a la identificación en la detección de baches en la malla vial de Bogotá.
- Resultados: Se observa el funcionamiento de lo implementado realizando un análisis detallado de cómo se aadecua la solución en la detección de baches en la malla vial de Bogotá.

Figura 1. Metodología del proyecto

METODOLOGÍA



Nota: Fases de la metodología para el desarrollo del proyecto. Elaboración propia.

Especificación del entorno del trabajo

Tabla 1. Especificación del entorno de trabajo

TIPO	P(DESEMPEÑO)	E(ENTORNO)	A(ACTUADORES)	S(SENSORES)
Sistema automático de detección de baches asistido por imágenes digitales.	Tasa de baches detectados vs total baches registrados, tiempo de respuesta anticipado	Vías, baches, automóvil con piloto	Notificación vía voz sintética del dispositivo móvil	Cámara, sensor de profundidad

Nota: Se especifica el entorno de trabajo de acuerdo a la descripción PEAS

Clasificación del entorno de trabajo

- Es parcialmente observable debido a que no se tiene acceso completo al entorno del sistema, solo se conocerá la captura de la imagen en tiempo real del sistema
- Su entorno de trabajo es monoagente porque no necesita de otros agentes para cumplir su objetivo.
- Estocástico porque su ambiente es cambiante, por lo tanto existen datos variables durante la ejecución del sistema.
- Secuencial porque los datos de imágenes obtenidos serán almacenados y dependen el uno del otro, para una posterior evolución del sistema, aprendiendo del nuevos datos recolectados.
- Dinámico porque el ambiente puede cambiar mientras el agente se encuentra en un estado de transición porque constantemente reajustará su clasificación en base a nuevo entrenamiento,
- Es continuo porque no existe un número limitado de posibles estados del ambiente, ya que tiene en cuenta las percepciones y acciones del sistema.

Datos

Para poder hacer un proceso de clasificación de imágenes es necesario tener un dataset con las imágenes relacionadas a lo que se quiere clasificar de manera automática. Para este caso trabajaremos con un dataset de 493 imágenes de baches bajadas de internet y otro dataset como

transformación de las 493 imágenes. A continuación se realiza una síntesis en relación a los datasets manejados en el proyecto.

Tabla 2. Reporte de características de los datasets

	Dataset 1	Dataset 2
Características de los datasets		
Tipo de archivos	.JPG y .JPEG	XML
Total de archivos	493	493
Tamaño de cada archivos	Desde 4 KB hasta 475 KB	1 KB- 2 KB
Características de los archivos	<p>Imágenes con:</p> <ul style="list-style-type: none"> • Diferentes ángulos de inclinación, tamaño, espacios, exposición de la luz y otros objetos que se encuentran dentro de la imagen. • Al igual que algunas imágenes repetidas pero alterando patrones como el Zoom para dar diferentes perspectivas(lejanía y cercanía),contraste, brillo de la imagen . <p>Todo esto para evitar el overfitting.</p>	Contiene las imágenes de dataset 1 pero en una resolución de 146 píxeles , convertidas en archivos XML con la respectiva etiqueta de “bache”.
Tamaño total del dataset	30.6 MB	304 KB

Materiales y Métodos

Lenguaje

Para el desarrollo del proyecto se optó por trabajar con el lenguaje Python ya que es un lenguaje con una sintaxis simplificada y nos permitirá trabajar con grandes listas de datos

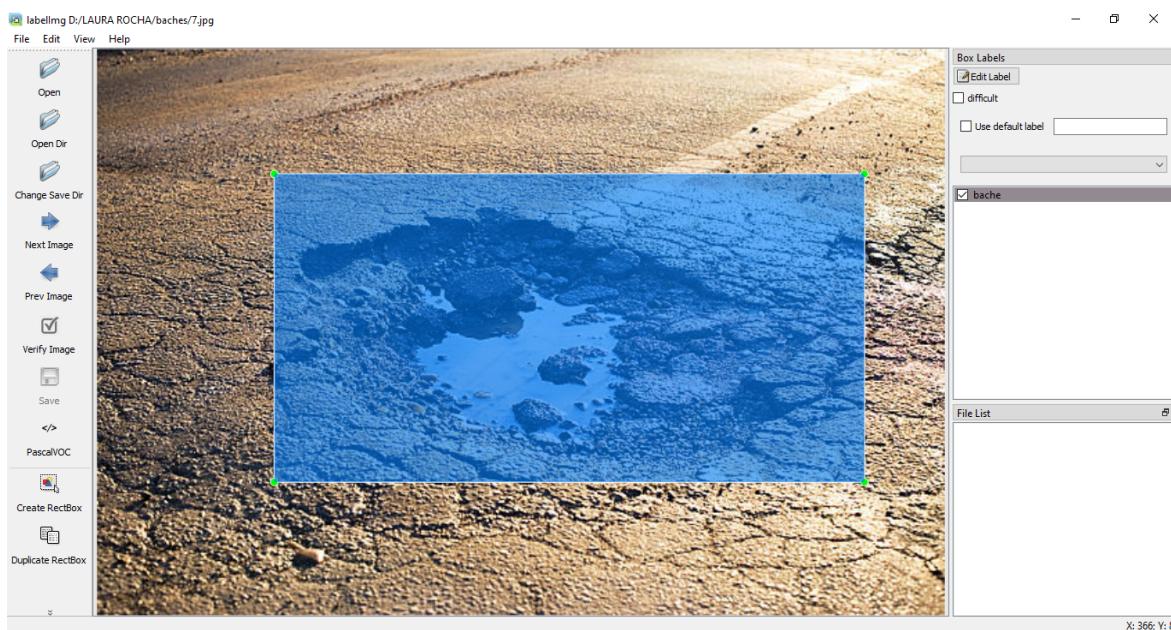
combinados de manera sencilla. Además tiene a su disposición muchas librerías enfocadas en el aprendizaje automático.

Entornos

Los entornos de desarrollo propuestos fueron Visual Studio Code, Jupyter Lab y Google Collab al final optamos por trabajar con Google collab ya que nos permitirá programar y correr el lenguaje Python además nos facilitará el trabajo de instalar las librerías necesarias para el desarrollo del proyecto.

Para la conversión de imágenes a formato XML con notaciones como la etiqueta que representa la clasificación de las imágenes se hizo uso de LabelImg como se observa en la figura 2 de forma manual seleccionando los espacios en las diferentes imágenes donde se encontraban los baches y asignando a estas un Label.

Figura 2. Herramienta LabelImg



Nota: imagen tomada de LabelImg utilizada para la conversión de la imagen a XML

Librerías y módulos

Las librerías y módulos que se utilizaron para trabajar fueron :

- **scikit-learn** que nos proporciona métodos para el aprendizaje automático.
- **Keras** nos ofrece la posibilidad de trabajar con redes de aprendizaje, numpy que nos ofrece la posibilidad de trabajar con grandes vectores y matrices y funciones matemáticas.
- **imgaug** que permitió agregar pequeñas alteraciones a las imágenes de entradas aumentando el dataset de imágenes y mejorando la capacidad de la red para detectar objetos.agregar desenfoque, agregar brillo, ó ruido aleatoriamente a las imágenes.
- **argparse** Este módulo genera automáticamente mensajes de ayuda ,uso y emite errores cuando se da como entrada al programa argumentos no válidos.
- **cv2** Ayudó en el análisis y procesamiento de las imágenes al igual que el seguimiento y detección de objetos.
- **JSON** Permite el intercambio de datos al momento de hacer uso del algoritmo YOLO
- **Copy** Permite crear copias de las imágenes contenidas en el dataset para realizar las alteraciones con la librería imgaug.

Algoritmo utilizado

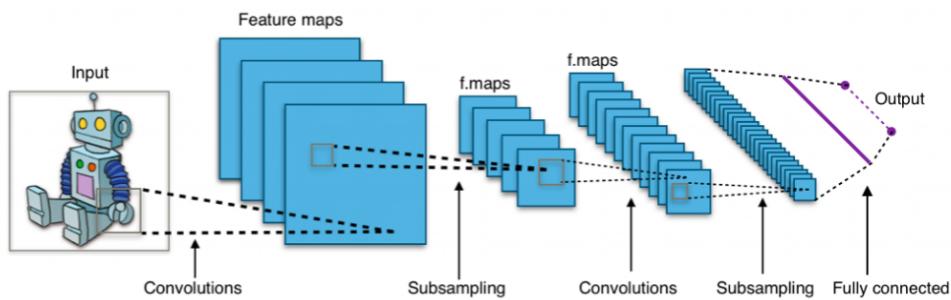
YOLO

El algoritmo “You Only Look Once” o mejor conocido por sus siglas YOLO es una red convolucional la cual permite la detección de objetos a partir del entrenamiento, esta se caracteriza por solo tener la necesidad de iterar un vez en la red creando la posibilidad de usar en máquinas que no tenga gran capacidad de procesamiento.

Cómo funciona

El algoritmo usa una red CNN que es una red que imita al ojo humano para identificar las características de las imágenes que son usadas como entradas para identificar objetos. Esta red tiene diferentes capas primero tenemos las que detectan líneas, curvas hasta llegar a las últimas capas que detectan formas mucho más complejas.

Figura 3. Funcionamiento de una red convolucional



Nota: Imagen tomada de

https://pochocosta.com/wp-content/uploads/2019/10/Typical_cnn-768x236.png

También se crea un grilla de 13 X 13 en cada una de las imágenes y sobre estas intenta detectar los objetos a partir de anclas fijas, se puede entender que si se toman 3 anclas con 3 tamaños diferentes y de esta manera podrá obtener como resultado 9 predicciones por cada celda. Para cada una de las celdas se dan valores entre 0 y 1

Figura 4. Grilla creada por YOLO para la predicción.



Nota: Imagen tomada del proceso realizado por YOLO en el dataset que se definió.

También usa una métrica de evaluación llamada IoU o Intersection over Union que se utiliza para medir la precisión de un detector de objetos con un conjunto de datos en particular. Para la generación de propuestas de clasificación de las regiones candidatas utiliza Non-Max-supression.

Figura 5. Métrica IoU para la precisión de la detección de objetos.

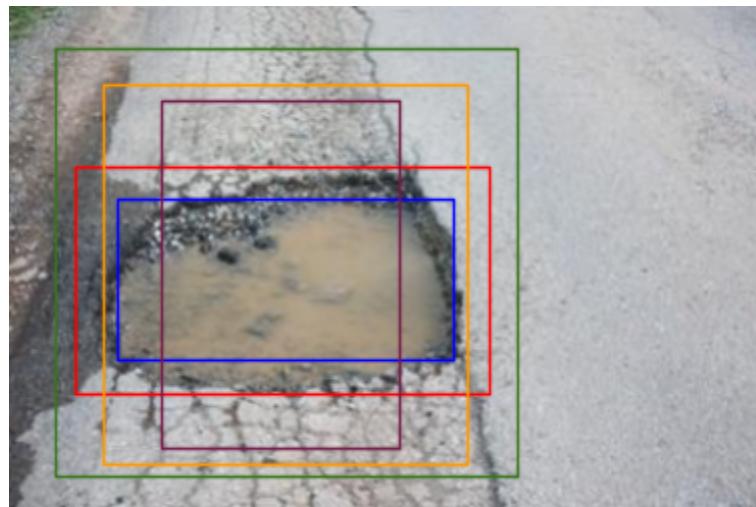
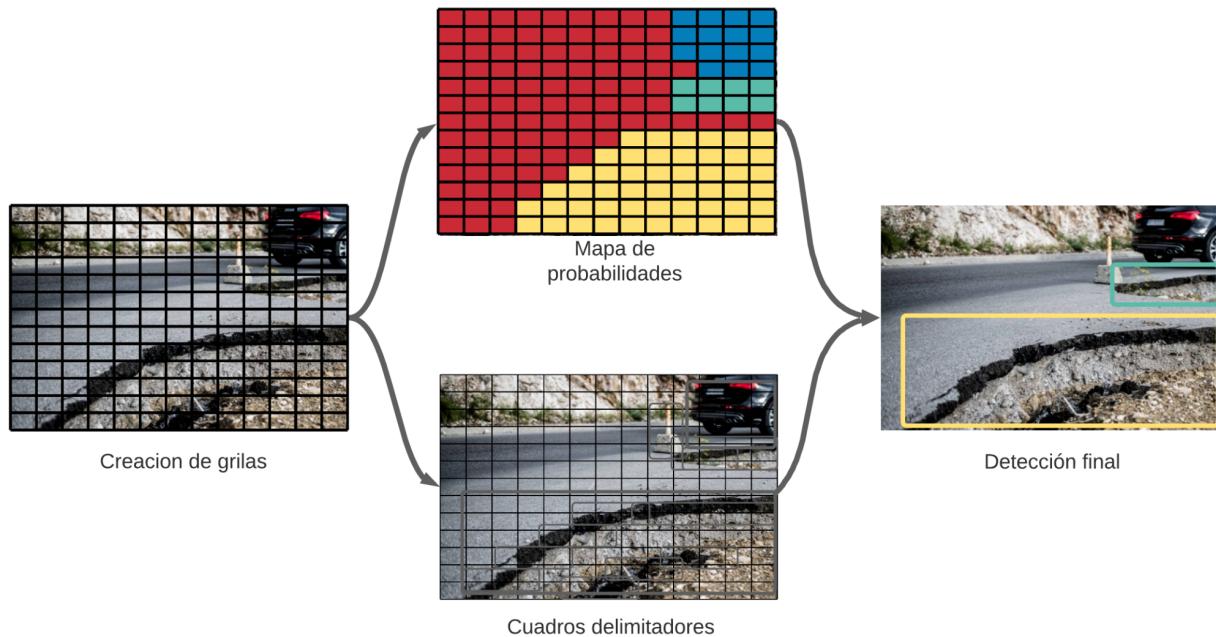


Figura 6. Procedimiento que sigue el algoritmo para etiquetar objetos.



Nota: Elaboración propia

Resultados

Como se menciona en la sección de materiales y métodos el sistema automático para la detección de baches en la malla vial de Bogotá asistido por imágenes digitales se realizó utilizando YOLO con 394 imágenes de entrenamiento y 99 para la validación, la red convolucional está compuesta por 22 capas convolucionales que crean los patrones que encontrará en las imágenes (en sus pixeles) para poder diferenciar entre los baches a clasificar. Del total parametros recibidos 50,578,686 fueron entrenables 50,558,014 parámetros y los datos que no se entrenaron 20,672 , haciendo uso de la GPU tardó 30 minutos en entrenar las 394

imágenes del dataset de bache con 6 épocas y 5 veces cada imagen con data augmentation, (en total se procesan 1970 imágenes en cada epoch).

Figura 7. Entrenamiento de la red neuronal convolucional.

```

Epoch 00001: val_loss improved from inf to 10.00909, saving model to red_bache.h5
WARNING:tensorflow:From /tensorflow-1.15.2/python3.7/keras/callbacks/tensorboard_v1.py:343:

Epoch 2/8
- 175s - loss: 10.0073 - val_loss: 10.0065

Epoch 00002: val_loss improved from 10.00909 to 10.00648, saving model to red_bache.h5
Epoch 3/8
- 175s - loss: 1.2543 - val_loss: 0.5424

Epoch 00003: val_loss improved from 10.00648 to 0.54240, saving model to red_bache.h5
Epoch 4/8
- 175s - loss: 0.7234 - val_loss: 1.8185

Epoch 00004: val_loss did not improve from 0.54240
Epoch 5/8
- 176s - loss: 0.4942 - val_loss: 0.5456

Epoch 00005: val_loss did not improve from 0.54240
Epoch 6/8
- 175s - loss: 0.4097 - val_loss: 0.9881

Epoch 00006: val_loss did not improve from 0.54240
Epoch 00006: early stopping
bache 0.0597
mAP: 0.0597

```

A partir de la figura 6 se realiza el respectivo análisis de los resultados obtenidos:

- val_loss: Porcentaje de pérdida, es variable y disminuye progresivamente a medida que se tiene mayor cantidad de imágenes, como se observa en la figura 6 baja el porcentaje significativamente al llegar a la sexta y última época con un valor de pérdida de 0.5424.
- mAP: Los resultados vienen dados por la métrica mAP que calcula la precisión promedio para cada clase en función de las predicciones con el bache Dataset se logró un 59.7 % de precisión.

A continuación en las figuras 7 y 8 observe la red con imágenes nuevas, y cómo se comporta la red.

Figura 7. Test de pruebas con imágenes aleatorias en diferentes ángulos



Nota: En la figura 7 se puede observar la clasificación exitosa del bache predicha por el modelo YOLO, adicionalmente se visualiza la métrica Métrica IoU identificando las zonas características de la imagen.

Figura 8. Test de pruebas 2 con imágenes aleatorias en diferentes ángulos



Nota: En la figura 8 se puede observar la clasificación del bache predicha por el modelo YOLO, el cual presenta fallos al enmarcar en cuadro verde únicamente el bache, posiblemente por la grieta del lado derecho. Adicionalmente se visualiza la métrica Métrica IoU identificando las zonas características de la imagen.

Discusión

Respecto al objetivo inicialmente planteado como expectativa de “Diseñar un prototipo funcional de un sistema para la detección de baches en la malla vial de Bogotá empleando tecnologías de visión por computador supervisado y técnicas propias del aprendizaje profundo como el entrenamiento y clasificación de imágenes ,el cual, capture imágenes, compare con el dataset dado, detecte a distancia y notifique al conductor la presencia de un bache mientras está conduciendo”. Se realizó el siguiente análisis de los resultados obtenidos:

En un primer momento se planteó usar un dataset ya creado pero debido a problemas de manejo del dataset citado en la propuesta metodológica, se optó por diseñar un set de datos propio con 493 imágenes y 493 xml de baches en las calles.

En primera instancia, se implementó un clasificador de imágenes el cual es capaz de clasificar los baches y por causas de tiempo no se le dio la posibilidad de clasificar los otros daños en la carretera como lo son los hundimientos y piel de cocodrilo.

Debido a que en la entrega anterior se tenían problemas con incompatibilidad con formatos como los png ya que creaba vectores más grandes que no permitían hacer la comparación con los otros vectores.Se procedió hacer uso de LabelImg para evitar presentar este tipo de errores y así mismo para asignar las diferentes etiquetas a las imágenes con mayor precisión. También se tuvo en cuenta el concepto de overfitting, procurando escoger imágenes con distintos ángulos, tamaños y tipos.

El modelo implementado está hecho para funcionar en dispositivos con pocos recursos o aplicaciones móviles ya que utiliza el algoritmo YOLO el cual es muy rápido por lo cual se

puede usar para la detección en tiempo real de objetos pero este tiene la desventaja de que sacrifica la precisión por la velocidad de procesamiento.

Finalmente nuestro prototipo tiene un 69.7 % de efectividad en la detección de baches, una medida de desempeño considerable si se considera que el dataset cuenta con pocas imágenes.

REFERENCIAS

- B. (2017, 8 marzo). Nueva estrategia para tapar 90.000 huecos en diez meses en Bogotá. *El Tiempo*.
- <https://www.eltiempo.com/bogota/plan-de-la-alcaldia-para-tapar-huecos-en-vias-de-bogota-64894>
- Galeano, I. A. S., & Maldonado, J. A. H. (2019). Detección de baches en vías urbanas a partir de imágenes de alta resolución espacial, mediante técnicas de GEOBIA. *Revista de Topografía AZIMUT, 10(1)*.
- Henriquez Rodríguez, M. J., Martínez Mendoza, H. G., & Martínez Mendoza, H. F. (2016). *Diseño e implementación de un software como solución para la optimización de los procesos de identificación, registro, seguimiento y control de los baches en la ciudad de Bogotá* (Bachelor's thesis, Universidad Piloto de Colombia).
- K, S. (2021, 30 abril). *Non-maximum Suppression (NMS) - Towards Data Science*. Medium.
- <https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c>
- Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., & Omata, H. (2018). Road damage detection using deep neural networks with images captured through a smartphone. *arXiv preprint arXiv:1801.09454*.
- Manzanares González, A. (2019). Detector de baches con deep learning.

N. (2020, 19 diciembre). *Modelos de Detección de Objetos*. Aprende Machine Learning.

<https://www.aprendemachinelearning.com/modelos-de-deteccion-de-objetos/>

Python. (s. f.). *3.9.5 Documentation*. Python Documentation. Recuperado 26 de mayo de 2021, de <https://docs.python.org/3/index.html>

Rojas Barreth, V. H. (2018). *Diseño de un sistema electrónico de detección de baches para asistencia vehicular* (Doctoral dissertation, Universidad de Guayaquil. Facultad de Ingeniería Industrial. Carrera de Ingeniería en Teleinformática.).

Rosebrock, A. (2021, 16 abril). *Intersection over Union (IoU) for object detection*.

PyImageSearch.

<https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>

Anexos

Anexo A. Código y dataset del proyecto [Clasificación de baches](#)