Journal of Research and Innovation in Civil Engineering
Vol. 1, N. 2, 18 September 2021

Detección automática del daño en estructuras de concreto reforzado usando redes neuronales

Automatic damage detection in reinforced concrete structures using neural networks

Alex Bryan Casilla Gallegos[a,*] & Wilder Ernesto Abad Jimenez[b]

a Escuela Profesional de Ingeniería Civil, Universidad de Ingeniería y Tecnología, Lima, Perú, alex.casilla@utec.edu.pe
b Escuela Profesional de Ingeniería Civil, Universidad de Ingeniería y Tecnología, Lima, Perú, wilder.abad@utec.edu.pe
** Autor de correspondencia*

Recibido: 18 agosto 2021 / Recibido en formato revisado: 10 setiembre 2021 / Aceptado para publicación: 16 setiembre 2021 / Disponible en línea 18 de setiembre 2021

Resumen
La detección de daño en estructuras de concreto es un aspecto crítico de los planes de gestión de la infraestructura civil. Este artículo propone un sistema económico de detección y clasificación de daños, fisuras y descascaramiento, en elementos de concreto reforzado. Principalmente, este sistema se basa en el uso de redes neuronales con el objetivo de realizar las mediciones automáticas y sin contacto para obtener una inspección más efectiva. El primer paso en el desarrollo del sistema propuesto es la colección y curación de la data. El segundo consiste en la preparación y etiquetado de la data. Posteriormente, la data se emplea para entrenar los algoritmos de aprendizaje profundo. Por último, los algoritmos entrenados son implementados en una interfaz gráfica que permite su uso intuitivo. A partir de la calibración del sistema con una serie de imágenes, se obtuvo un 98% de precisión para Unet y 74.85% para Yolov3.
Palabras clave: redes neuronales; concreto reforzado; detección; clasificación.

Abstract
Detecting damage to concrete structures is a critical aspect of civil infrastructure management plans. This article proposes an inexpensive system for detecting and classifying damage, cracks and spalling, in reinforced concrete elements. Mainly, this system is based on the use of neural networks in order to carry out automatic and non-contact measurements to obtain a more effective inspection. The first step in the development of the proposed system is the collection and curation of the data. The second consists of the preparation and labeling of the data. Subsequently, the data is used to train deep learning algorithms. Finally, the trained algorithms are implemented in a graphical interface that allows their intuitive use. From the calibration of the system with a series of images, a 98% precision was obtained for Unet and 74.85% for Yolov3.
Keywords: *Neural networks; reinforced concrete; detection; classification.*

1 Introducción

El uso extensivo de concreto reforzado en las edificaciones del Perú hace necesario el desarrollo de nuevos métodos de reconocimiento de patologías para su tratamiento y prevención. Estas patologías pueden ser fisuras, grietas, desmoronamientos, cangrejeras y entre otras, teniendo como origen el mezclado, vaciado, vibrado, cargas extremas, fallas de diseño o agentes ambientales como el salitre, humedad, etc.

Para el caso de las fisuras y grietas se encontraron 2 tipos de clasificación, la primera se basa en las dimensiones de las patologías, específicamente basándose en el ancho (e) y en el segundo caso, se puede dividir aún más las patologías según el origen y/o causa de las grietas, en este caso solo utilizaremos las grietas más comunes encontradas en el concreto [1].

Tabla 1.
Clasificación de fisuras y grietas mediante su ancho (e).

Clasificación por ancho (e)		Nivel de repercusión en la estructura
Microfisuras	e < 0.05mm	Nivel muy bajo.
Fisuras	0.1<e<0.2mm	Nivel bajo. Vulnerable a ambiente marinos u otros de naturaleza agresiva que podrían dañar las varillas refuerzos.
Microfisuras	0.2<e<0.4mm	Nivel moderado. Posible presencia de daños estructurales, se requiere de una investigación más a fondo para determinar el daño y el posible método de reforzamiento y/o reparación
Grietas	0.4<e<1.0mm	Nivel alto. Podría existir reducción de la capacidad sismorresistente. Es necesario un estudio de vulnerabilidad para el diagnóstico, y alternativas de reparación y/o reforzamiento.
	e>1.0mm	Nivel muy alto. Posible reducción significativa de la capacidad sismorresistente. Se requiere estudio de vulnerabilidad para el diagnóstico y determinar la posibilidad de salvar la estructura. Dependiendo de los daños encontrados, se debe evaluar la evacuación y apuntalamiento de la edificación.

Fuente: [1] Sotomayor, 2018.

Tabla 2.
Identificación de fisuras y grietas respecto a su forma.

Clasificación por ancho (e)		Nivel de repercusión en la estructura
Microfisuras	e < 0.05mm	Nivel muy bajo.
Fisuras	0.1<e<0.2mm	Nivel bajo. Vulnerable a ambiente marinos u otros de naturaleza agresiva que podrían dañar las varillas refuerzos.
Microfisuras	0.2<e<0.4mm	Nivel moderado. Posible presencia de daños estructurales, se requiere de una investigación más a fondo para determinar el daño y el posible método de reforzamiento y/o reparación
Grietas	0.4<e<1.0mm	Nivel alto. Podría existir reducción de la capacidad sismorresistente. Es necesario un estudio de vulnerabilidad para el diagnóstico, y alternativas de reparación y/o reforzamiento.
	e>1.0mm	Nivel muy alto. Posible reducción significativa de la capacidad sismorresistente. Se requiere estudio de vulnerabilidad para el diagnóstico y determinar la posibilidad de salvar la estructura. Dependiendo de los daños encontrados, se debe evaluar la evacuación y apuntalamiento de la edificación.

Fuente: [1] Sotomayor, 2018

Algunos de los métodos usados para la detección de patologías en concreto son el método VPU que se basa en la determinación de la propagación de un pulso ultrasónico a través de un material [2]. Otro método de detección de patologías es el ensayo de líquidos penetrantes que es un ensayo no destructivo que permite identificar discontinuidades o grietas en el concreto.
Estos métodos mencionados son efectivos, sin embargo, implican un mayor uso de materiales, inversión, tiempo y mano de obra que limita su aplicación continua en construcciones. Por ello, se plantea un sistema de detección económico y automático, con el uso de redes neuronales, que permita el reconocimiento de patologías y su clasificación para una posterior evaluación de riesgos. El tipo de medición de este sistema es del tipo sin contacto, es decir, no será invasiva.
El alcance de este trabajo de investigación se acerca más al desarrollo de nuevas tecnologías que permitan detectar y monitorear las estructuras de una manera rápida, simple y económica, a través del uso de redes neuronales e imágenes que pueden ser fácilmente usadas. Se busca, entonces, desarrollar métodos cada vez más eficaces y a la vez fáciles de usar que permitan una detección temprana de fallas en las estructuras, consiguiendo así, prevenir costos de reparación o aún peor, desastres.

2 Materiales y métodos

Dentro de la investigación se buscará analizar una gran variedad de alternativas de redes neuronales y seleccionar la que mejor se adapte a la aplicación que le estamos dando; en segundo lugar, se buscará lograr entrenar el modelo para identificar hasta 2 tipos de patologías, en este caso grietas y descascaramientos y en tercer lugar, será necesario realizar el desarrollo de Interfaz ejecutable para el reconocimiento y clasificación.

Con respecto a los dos primeros focos: clasificar e identificar patologías en concreto, grietas y descascaramiento. Se emplearon las redes neuronales de Yolov3 y U-Net en el entorno de Google Colab.

Yolov3

Es un sistema de detección con herramientas pre entrenadas de detección de objetos en imágenes y vídeos en tiempo real. Es una sola red convolucional que predice simultáneamente múltiples cuadros delimitadores. Este sistema divide la imagen en cuadrículas de N x N, para así, si un objeto se ubica en el centro de una de las cuadrículas, esta cuadrícula será la encargada de detectar ese objeto. Yolov3 es entrenado con imágenes completas para optimizar el rendimiento de detección.[2]

En la siguiente figura, podemos apreciar la arquitectura de Yolov3, esta se le conoce como DarkNet-53 la cual consta de 53 capas convolucionales y tiene como función principal la extracción de características importantes. Por cada convolución se realiza un *batch normalization*, que asegura la normalización de la data en todo instante.[3]

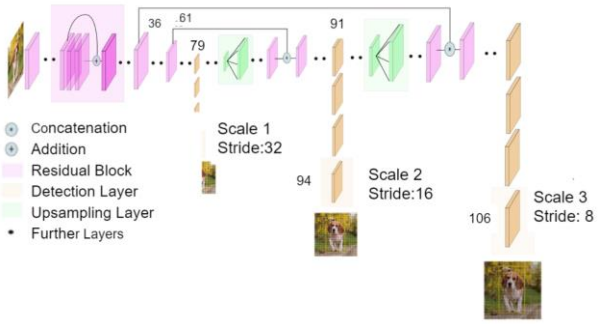


Figura 1. Arquitectura de Yolov3.

U-net

Es una red convolucional, es decir es un algoritmo cuya función principal es clasificar imágenes basándose en imágenes ya clasificadas, estas tienen que ser proporcionadas a la red de manera manual.

Las redes convolucionales trabajan en etapas, en la primera se tiene la capa convolucional, aquí es donde se tiene la imágen base como entrada y se pasa por un filtro que cubre un área de la imagen y chequea si el objeto está presente, finalmente compara el filtro con la imagen original para detectar las partes del objeto que estamos buscando.

En la segunda etapa se tiene el submuestreo, mediante el uso de un filtro se va seleccionando los pixeles que son más probables en contener la imagen y la bota en una imagen de una dimensión menor a la original. En la tercera etapa es en la que ocurre la clasificación, basándose en la información de las etapas y filtros anteriores. [4]

Lo que vuelve único a U-net es que además de tener este lado de compresión visto en la Figura 2 en la parte de la izquierda. Es que también tiene el lado derecho que ocurre de forma simétrica pero ahora lo expande, lo que hace es concatenar la imagen cortada con la imagen original. [5]

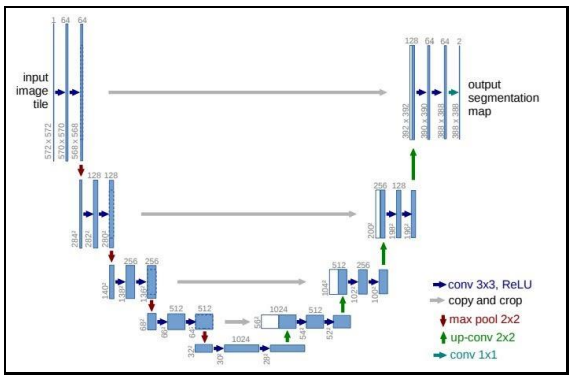


Figura 2. Esquema de U-net.

En general, los pasos a seguir en ambos casos son: colección de datos, preprocesamiento (cambio de escala, contornos, etc) y etiquetado, entrenamiento, testeo e implementación del sistema propuesto.

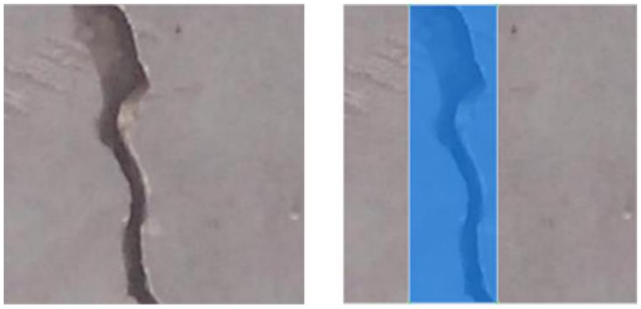
Colección de datos

Dentro de la colección de datos definimos las fuentes de extracción de la dataset. La dataset empleada en el entrenamiento de la red neuronal Yolov3 consiste en 1241 imágenes, en formato jpg, de grietas y descascaramiento. Estas imágenes fueron extraídas de las investigaciones *Deep Concrete Inspection Using Unmanned Aerial Vehicle Towards CSSC Database* [6] y *Road Crack Detection Using Deep Convolutional Neural Network and Adaptive Thresholding* [7]. Por otro lado, la dataset de U-Net se obtuvo de *Comparison of crack segmentation using digital image correlation measurements and deep learning* [8]. Se debe resaltar que la cantidad de imágenes extraída para esta red fue de 301 imágenes en formato jpg.

Etiquetado del dataset:

Posteriormente, es necesario realizar un etiquetado al dataset de imágenes recaudadas para realizar el entrenamiento de los modelos. En este caso, dicho proceso varía dependiendo de la red neuronal a entrenar.

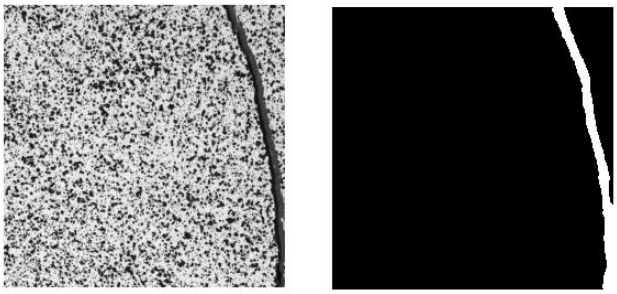
En el caso de Yolov3 las imágenes son procesadas en la plataforma *LabelImg*. Dicha plataforma permite enmarcar las fisuras de cada imagen y almacenar sus coordenadas en un formato .txt, el cual es requerido por Yolov3 para su entrenamiento.



(a) (b)

Figura 3. Imagen etiquetada para Yolov3 (a) Imagen real (b) Etiqueta de la imagen.

Por otro lado, las etiquetas para U-net son imágenes que muestran la forma de la grieta en color blanco mientras que lo demás elementos que no son grietas se cubren totalmente de negro, un ejemplo de ello es el mostrado en la Figura 4. Este proceso se puede realizar en cualquier plataforma de edición de imágenes como Photoshop o GIMP, sin embargo, en este caso el dataset usado incluía las etiquetas para cada imagen.



(a) (b)

Figura 4. Imagen etiquetada para U-net (a) Imagen real (b)Etiqueta de la imagen.

Entrenamiento

Para el proceso de entrenamiento se empleó en ambos casos el entorno de Colaboratory, también llamado Colab, ya que permite escribir y ejecutar código de Python en un navegador aprovechando el hardware de Google, específicamente la unidad de procesamiento gráfico (GPU). Se debe resaltar que otro beneficio de este entorno es el trabajo colaborativo y el acceso a Drive.

Desde el proceso de entrenamiento se emplean bibliotecas particulares de aprendizaje profundo para cada red neuronal, con el fin de realizar cálculos numéricos haciendo uso de la programación de tensores. En el caso de Yolov3 se emplea *PyTorch* creado por el laboratorio de investigación de inteligencia artificial de Facebook; mientras que U-net emplea *TensorFlow*, el cual fue desarrollado por Google.

Testeo

Al finalizar la etapa de entrenamiento, es necesario realizar un test de validación del modelo. El test consiste en seleccionar algunas imágenes y realizar una predicción de resultados usando la versión del modelo ya entrenado.

Se realizó una predicción de resultados tanto para Yolov3 como Unet y se recaudaron algunas métricas de desempeño como; "Accuracy" la cual indica qué tan precisa es la predicción del modelo comparado a sus etiquetas. Asimismo, la métrica "loss" indica cuánto le falta al modelo para llegar a su límite de aprendizaje.

Implementación del sistema propuesto

Como se explicó cada red neuronal tiene una funcionalidad en específica, clasificar o identificar. No

obstante, el sistema propuesto busca unificar la capacidad de ambas redes neuronales, por ello se desarrollará una interfaz para que el usuario pueda ejecutar ambos códigos. Para el desarrollo de la interfaz emplearemos la librería *tkinter*, el módulo integrado en python para crear este tipo de aplicaciones.

3 Resultados

Clasificación de daños

El entrenamiento en Yolov3 con las 1241 imágenes duró 6 horas 10 minutos y 30 segundos. A partir del entrenamiento se obtuvo el archivo entrenado, denominado dentro de la codificación como yolov3_ckpt_199.pth. A continuación, se muestran las métricas del archivo entrenado:

Tabla 3.
Evaluación de modelo - Yolov3.

Index	Class name	AP	mAP
0	crack	0,74254	0,74853
1	spalling	0,75452	

Fuente: Elaboración propia.

Tabla 4.
Resultados del entrenamiento en la última época - Yolov3.

Metrics	Yolo Layer 0	Yolo Layer 1	Yolo Layer 2
grid_size	11	22	44
loss	0,602998	0,25501	0,209287
x	0,009416	0,053836	0,045239
y	0,006179	0,004038	0,024434
w	0,0051	0,001687	0,002105
h	0,005903	0,002253	0,001711
conf	0,576232	0,193064	0,135312
cls	0,000169	0,000132	0,000486
cls_acc	100%	100%	100%
recall50	0,75	1	1
recall75	0,75	1	1
precision	0,6	0,5	0,285714
conf_obj	0,797489	0,955794	0,963326
conf_noobj	0,001344	0,000757	0,000473
Total loss		1,067294836	

Fuente: Elaboración propia

En la Tabla 3, se observa que la métrica de precisión tiene 74.25% para crack; mientras que la de spalling tiene una precisión de 75.45%. De tal forma, la precisión promedio del modelo es 74.85%. Además, en la Tabla 4 se evidencia que Layer 1 y Layer 2 tienen un comportamiento aceptable, a diferencia del Layer 3. Esta variación sobre el comportamiento de los layers establece que el modelo tiene mayor capacidad de detección con objetos grandes y medianos. En general, las métricas nos brindan una percepción de efectividad en la clasificación de daños. No obstante, en futuros entrenamientos se debe buscar incrementar la cantidad de imágenes usadas con el fin de tener mejores resultados. Por último, empleamos el archivo entrenado junto al código de ejecución para visualizar la clasificación entre spalling y crack.

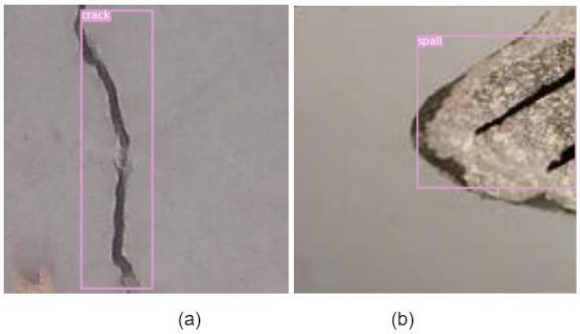


Figura 5. Clasificación por Yolov3: (a) Detección de crack -grieta (b) Detección de Spalling - Descascaramiento.

Segmentación de daños

El entrenamiento de U-net se realizó con 301 imágenes y sus respectivas etiquetas, dicho proceso duró aproximadamente 7 horas 8 minutos y 16 segundos. Durante este tiempo, el modelo realizó 17 epochs y se obtuvieron algunas métricas de rendimiento usando Tensorboard.

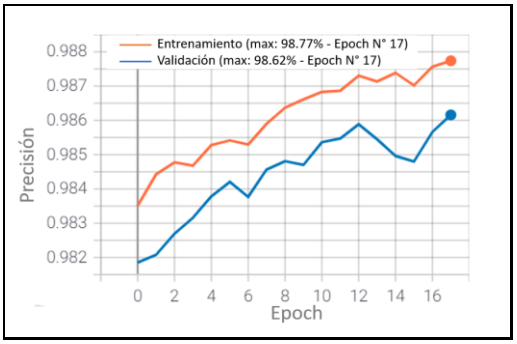


Figura 6. Resultados del entrenamiento - U-net: (a) Gráfica de precisión vs epoch (b) Gráfica de Loss vs epoch.

Como se muestra en la Figura 6, el modelo logró alcanzar una precisión de 98.77% en el proceso de entrenamiento y 98.62% durante la validación del modelo. Asimismo, se puede observar que en la Figura 6 (a) que existe una relación directamente proporcional entre la precisión y los epochs. Por ello, se cree que, usando un número mayor de imágenes y etiquetas en el entrenamiento, dicho modelo podría mejorar.

Finalmente se graficaron los resultados obtenidos en la segmentación se muestra en la Figura 7:

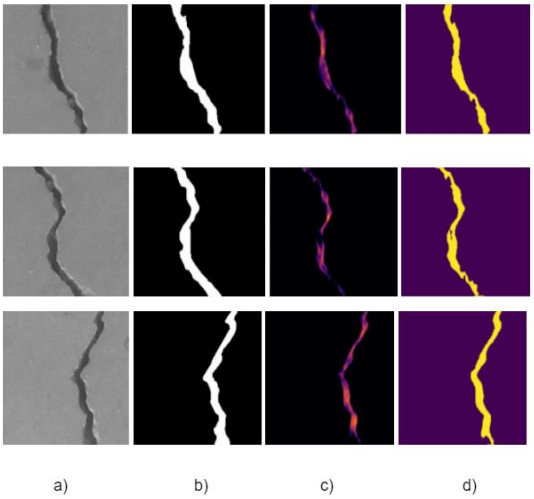


Figura 7. Segmentación por U-net: (a)Imagen real (b) resultado esperado (c)Predicción de U-net (d)Imagen de predicción uniformizada.

Implementación

La Figura 8 muestra el diagrama de flujo que explica la interacción del usuario con el sistema a emplear. Al inicio el usuario debe elegir si desea clasificar, identificar daños o ambos; dependiendo de la elección se hará uso de cada uno de los algoritmos. Una vez se le indique esto al programa, el usuario deberá colocar la ubicación de la fotos que desee analizar y el lugar donde serán guardadas las fotos ya analizadas. Terminado este paso se daría una vista previa de los resultados de las imágenes analizadas para después preguntarle al usuario si desea continuar trabajando con las mismas fotos o agregar nuevas, si la respuesta es no el programa termina.

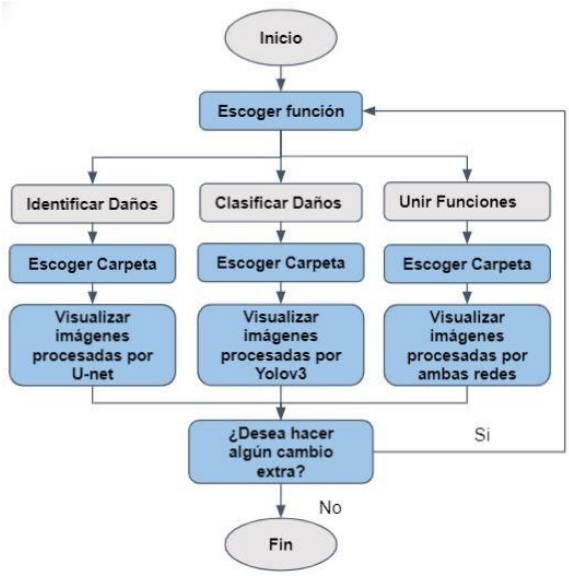
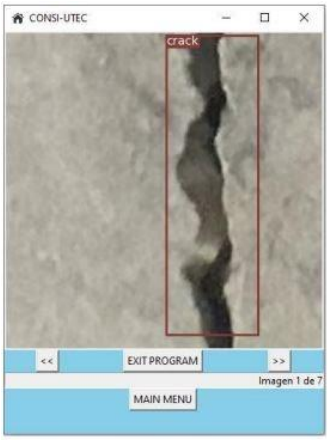


Figura 8. Diagrama de flujo de la interfaz del sistema de detección.

Con el diagrama queremos expresar nuestro deseo de hacer que este sistema sea interactivo y fácil de utilizar. En la Figura 9 se observa dicho diagrama en su primera versión, elaborado con la librería de tkinter. En esta versión no hay uso de widgets que le den detalles estéticos, sino solo la forma visual de la funcionalidad del sistema.

(a)



(b)

Figura 9. Implementación del sistema automático de detección:(a) Cuadro inicial (b) Imagen clasificada con YoloV3.

5 Conclusiones

El sistema mostró ser simple y efectivo para los objetivos del proyecto, por ejemplo, se obtuvo un 98% de precisión para Unet y 74.85% de precisión para Yolov3. Además, el sistema de implementación unió efectivamente las funcionalidades de cada modelo, logrando que tenga un uso intuitivo para todos los posibles usuarios del sistema.

Se logró identificar distintos tipos de patologías, que pueden ser clasificadas según distintos criterios. Por ello, es importante mencionar que una de las limitaciones de nuestro sistema propuesto radica en que algunas patologías conllevan mucha incertidumbre en su predicción y en estos escenarios, los modelos propuestos no podrían ser efectivos.

Si bien es cierto que se logró obtener buenos valores de precisión para nuestros modelos, estos aún no se han probado el sistema en condiciones con sombras o poca iluminación. Por ello, se recomienda evitar estas condiciones o estudiar una mejora para que las predicciones mejoren incluso en condiciones no óptimas.

Durante el periodo de entrenamiento, se pudo evidenciar que los requerimientos de hardware son muy exigentes, por ejemplo, es necesario usar una tarjeta gráfica para reducir el tiempo de entrenamiento. Por ello, se debe considerar que el entrenamiento de cada modelo implica tener un gabinete que pueda llevar el entrenamiento en un tiempo relativamente corto. De no ser el caso, se recomienda usar el entorno COLAB que brinda acceso remoto a algunos servidores de google con propósitos académicos.

Finalmente, podemos decir que la propuesta sirve como base para el monitoreo rápido y sin contacto de estructuras de concreto reforzado. Ello implica un incremento en la vida útil de algunas infraestructuras o la reducción del riesgo que podría correr el personal de monitoreo en áreas de difícil acceso.

6 Referencias bibliográficas

[1] Sotomayor C. Entendiendo a las fisuras y grietas en las estructuras de concreto.Consultcreto.com.2018. Disponible en: http://www.consultcreto.com/pdf/entendiendo.pdf

[2] Lorenzi A, Argenta Chies J, Santos Adamatti D, Pinto da Silva Filho LC. Evaluación de la capacidad de detección de fallas en concreto a través del ensayo ultrasónico. Rev ALCONPAT. 2017;7(3):286–301.

[3] De Palma R. YOLOv3 Tutorial: Understanding What is YOLOv3 and How it works? . Com.au. 2020 . Disponible en: https://bestinau.com.au/yolov3-architecture-best-model-in-object-detection/

[4] Miranda Pérez R, Solano Arias J, Méndez Porras A. Introducción al Aprendizaje Automático con YOLO. 6th ed. Guayaquil: Facultad de Ingenierías y Tecnologías de la Información y Comunicación; 2019. Disponible en: https://revistas.ulatina.ac.cr/index.php/tecnologiavital/article/view/250/261.

[5] IBM Cloud Education. What are Convolutional Neural Networks? Ibm.com. 2021 Disponible en: https://www.ibm.com/cloud/learn/convolutional-neural-networks

[6] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. Arxiv.org. 2018. Disponible en: https://arxiv.org/pdf/1505.04597.pdf

[7] Yang L, Li B, Li W, Zhaoming L. Deep Concrete Inspection Using Unmanned Aerial Vehicle Towards CSSC Database . Researchgate.net. 2017. Disponible en: https://www.researchgate.net/publication/319333841_Deep_Concrete_Inspection_Using_Unmanned_Aerial_Vehicle_Towards_CSSC_Database

[8] Fan R, Bocus MJ, Zhu Y, Jiao J, Wang L, Ma F, et al. Road crack detection using deep convolutional neural network and adaptive thresholding. In: 2019 IEEE Intelligent Vehicles Symposium (IV). IEEE; 2019. p. 474–9.

[9] Rezaie A, Achanta R, Godio M, Beyer K. Comparison of crack segmentation using digital image correlation measurements and deep learning. Constr Build Mater. 2020;261(120474):120474.