UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

IDENTIFICACIÓN AUTOMÁTICA DE LAS GRIETAS EN PISTAS DE ASFALTO UTILIZANDO PROCESAMIENTO DIGITAL DE IMÁGENES

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

Autor:

Bach. Ignacio Soto Percy Robustiano
ORCID: https://orcid.org/0000-0001-6304-7058

Asesor:

Dr. Tuesta Monteza Victor Alexci
ORCID: https://orcid.org/0000-0002-5913-990x

Línea de Investigación:

Infraestructura, Tecnología y Medio Ambiente

Pimentel – Perú 2022

APROBACIÓN DEL JURADO

IDENTIFICACIÓN AUTOMÁTICA DE LAS GRIETAS EN PISTAS DE ASFALTO UTILIZANDO PROCESAMIENTO DIGITAL DE IMÁGENES

_____

Bach. Ignacio Soto Percy Robustiano
Autor

_____

Dr. Tuesta Monteza Víctor Alexci
Asesor

_____ _____

Mg. Mejia Cabrera Heber Ivan
Presidente de Jurado

Mg. Cachay Maco Junior Eugenio
Secretario de Jurado

_____

Dr. Tuesta Monteza Víctor Alexci
Vocal de Jurado

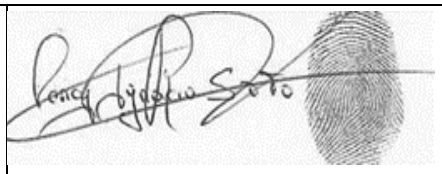
DECLARACIÓN JURADA DE ORIGINALIDAD

Quien(es) suscribe(n) la **DECLARACIÓN JURADA**, soy(somos) **egresado (s)** del Programa de Estudios de **Ingeniería de Sistemas** de la Universidad Señor de Sipán S.A.C, declaro (amos) bajo juramento que soy (somos) autor(es) del trabajo titulado:

IDENTIFICACIÓN AUTOMÁTICA DE LAS GRIETAS EN PISTAS DE ASFALTO UTILIZANDO PROCESAMIENTO DIGITAL DE IMÁGENES

El texto de mi trabajo de investigación responde y respeta lo indicado en el Código de Ética del Comité Institucional de Ética en Investigación de la Universidad Señor de Sipán (CIEI USS) conforme a los principios y lineamientos detallados en dicho documento, en relación a las citas y referencias bibliográficas, respetando al derecho de propiedad intelectual, por lo cual informo que la investigación cumple con ser inédito, original y autentico.

En virtud de lo antes mencionado, firman:

IGNACIO SOTO, PERCY ROBUSTIANO	43348579	

Pimentel, 30 de Enero de 2023.

Dedicatorias

La presente investigación se la dedico a Dios por otorgarme buena salud.

A mis Padres Percy Ignacio Marchena y Angélica Soto Milián y a mis hermanos por su amor infinito

A mi asesor Mg. Víctor Alexci Tuesta Monteza por la contribución para la finalización de la investigación.

Agradecimientos

"No tengo talentos especiales, pero sí soy profundamente curioso"
- Albert Einstein.

Agradezco de todo corazón a Mis Maestros de la Universidad Señor de Sipán por brindarme todo su apoyo desinteresado muchas gracias y bendiciones para ustedes y toda su familia.

Resumen

En nuestro País las pistas de asfalto en su gran mayoría se encuentran en muy mal estado. Dónde el salitre, las lluvias, las malas prácticas en construcción de pistas de asfalto son las causas principales que dan origen a las grietas en pistas de asfalto. En la actualidad una de las tendencias en las ciencias computacionales es el desarrollar técnicas que faciliten la información para su interpretación en el procesamiento digital de imágenes, ya que si se generan técnicas más eficientes es posible aumentar la exactitud y disminuir la complejidad computacional. Existen otras técnicas actuales que se han generado en investigación para obtener la segmentación como los algoritmos de máxima entropía, el método otsu, el crecimiento por regiones, el umbral global iterativo, los cuales ofrecen mejores resultados para segmentar en distintos niveles, cuya función principal es hacer crecer los pixeles iterativamente utilizando comparaciones entre pixeles con una medida de similitud. Pese al potencial de estos algoritmos de segmentación, tienen una gran cantidad de desventajas al aplicarlos a la carretera. Debido a los entornos dinámicos, iluminación variante y vibración, los algoritmos tienen desventaja porque son muy sensibles al ruido, dependen del punto de inicialización para generar una buena o mala segmentación y al ser iterativos tienden a consumir mucho recurso computacional. La presente investigación, pretende proporcionar una técnica para la realización de procesos cuyo resultado tiene una implicación importante en el proceso digital de imágenes. La construcción del brazo metálico, me permitió captar las imágenes de grietas en pistas de asfalto. El algoritmo de Red Neuronal Convolucional (CNN) tuvo buena performance en los trabajos de investigación relacionados en la clasificación de grietas, por tal motivo se escogió este algoritmo. El procesamiento de las imágenes, aumentando el brillo y aplicando el filtro umbralización binario, permitió visualizar mejor las grietas de asfalto y esto mejoró la identificación del modelo. La realización de la propuesta demuestra que de los resultados alcanzados para el algoritmo de CNN obtuvo 98% de exactitud en tiempo promedio de 85 segundos

Palabras clave: Grietas, suprimir ruido, asfalto, arreglo de contraste, ratificación de intensidad.

Abstrac

In our country, the vast majority of asphalt tracks are in very poor condition. Where the saltpeter, the rains, the bad practices in the construction of asphalt tracks are the main causes that give rise to cracks in asphalt tracks. Currently, one of the trends in computational science is to develop techniques that facilitate information for its interpretation in digital image processing, since if more efficient techniques are generated, it is possible to increase accuracy and reduce computational complexity. There are other current techniques that have been generated in research to obtain segmentation, such as maximum entropy algorithms, the otsu method, growth by regions, the iterative global threshold, which offer better results to segment at different levels, whose main function is grow pixels iteratively using comparisons between pixels with a measure of similarity. Despite the potential of these segmentation algorithms, they have a large number of disadvantages when applied to the road. Due to dynamic environments, varying lighting and vibration, the algorithms have a disadvantage because they are very sensitive to noise, they depend on the initialization point to generate a good or bad segmentation and, being iterative, they tend to consume a lot of computational resources. This research aims to provide a technique for carrying out processes whose result has an important implication in the digital image process. The construction of the metal arm allowed me to capture the images of cracks in asphalt tracks. The Convolutional Neural Network (CNN) algorithm had a good performance in research works related to crack classification, for this reason this algorithm was chosen. The processing of the images, increasing the brightness and applying the binary thresholding filter, allowed better visualization of the asphalt cracks and this improved the identification of the model. The realization of the proposal shows that the results achieved for the CNN algorithm obtained 98% accuracy in an average time of 85 seconds.

Keword: Crack detection, suppress noise, asphalt, contrast arrangement, intensity ratification.

Índice

I. INTRODUCCIÓN 10

1.1. Realidad Problemática 10

1.2. Trabajos previos 12

1.3. Teorías relacionadas al tema 17

1.3.1. Pistas asfaltadas 17

1.3.2. Clasificación no Supervisada 19

1.3.3. Métodos ó técnicas para identificar grietas 20

1.3.4. Imágenes 22

1.3.5. Procesamiento de imágenes 27

1.3.6. Clasificadores 33

1.3.7. Lenguaje de programación Python 38

1.4. Formulación del Problema 40

1.5. Justificación e importancia del estudio 40

1.6. Hipótesis 41

1.7. Objetivos 41

1.7.1. Objetivo general 41

1.7.2. Objetivos específicos 41

II. MÉTODO 42

2.1. Tipo y Diseño de Investigación 42

2.1.1. Tipo de Investigación 42

2.1.2. Diseño de Investigación 42

2.2. Población y muestra 42

2.2.1. Población 42

2.2.2. Muestra 42

2.3. Variables, Operacionalización 42

2.3.1. Variable independiente 42

2.3.2. Variable dependiente 42

2.3.3. Operacionalización de variables 43

2.4. Técnicas e instrumentos de recolección de datos, validez y confiabilidad. 44

2.5. Procedimiento de análisis de datos 45

2.6. Criterios éticos 45

2.7. Criterios de Rigor Científico 45

III. RESULTADOS 46

3.1. Resultados en Tablas y Figuras 46
3.2. Discusión de resultados 50
3.3. Aporte práctico. 51
3.3.1. Establecimiento del protocolo de adquisición de imágenes de grietas de pistas de asfalto. 53
3.3.2. Selección de los algoritmos de clasificación 56
3.3.3. Adquisición de las imágenes de grietas de asfalto 58
3.3.4. Implementación del algoritmo seleccionado 61
3.3.5. Análisis de los resultados obtenidos 75
IV. CONCLUSIONES Y RECOMENDACIONES 77
4.1. Conclusiones 77
4.2. Recomendaciones 77
ANEXOS 82

I. INTRODUCCIÓN

1.1. Realidad Problemática

Hace cincuenta años la mayoría de nuestras pistas no estaban asfaltadas en su totalidad. Por lo general, el estado en que se encontraban era intransitable a causa del salitre, lluvias, malas prácticas, circulación del tránsito pesado. Sin embargo; el último estudio que reporta el Ministerio de Transportes y Comunicaciones informa que el 86% de nuestras pistas ya se encuentran asfaltadas (MTC, 2016). La Cámara de Comercio de Lima nos informa que las pistas asfaltadas a nivel nacional se encuentran deterioradas a gran escala. En el Perú, las eventualidades de tráfico terrestres son las principales incógnitas que afronta y preocupa a nuestra nación. Según lo publicado en el diario el correo, el problema radica en el mal estado que se encuentran las pistas. (Diario el correo, 2019).

Si se produjera una desgracia, la existencia de nuestros semejantes sólo se valdrían de las correas de efectividad y de los sacos de aire, este último; sí las tuviese. Sin embargo, las altas velocidades y en pistas deterioradas, ejercen que estos mecanismos de confianza sirvan muy poco para minorar una catástrofe. (La república, 2019).

Un indicador importante del estado de las pistas de asfalto es la exhibición de rajaduras, ya que provocan múltiples perjuicios. De modo, que revelar las rajaduras en pistas de asfalto es decisivo para impedir y poder conservarlas. En el Perú, los modelos de evaluación manuales más utilizados son: la inspección de pavimentos de la Superintendencia de Transporte Terrestre (SUTRAN, 2016), el indicativo de Circunstancia de Pavimentos y el Programa de Investigación Estratégico de Carreteras (Revista CUC, 2016). Sin embargo, ya se están integrando tecnologías para contribuir a la gestión y a la intervención de carreteras; además, en el ámbito mundial, se están optando nuevas tecnologías para percibir y distinguir la referencia de modo automático o semiautomático. (Townes, 2014). La inspección visual de pavimentos consiste contratar personas para que llenen los formatos a mano. Este procedimiento, es pausado, peligroso y de elevado precio en la contratación de personal. El componente que permite disminuir el grado de incertidumbre es la estandarización y sistematización del desarrollo de supervisión.

La opción para obtener esto, es a través del procedimiento de figuras, ya que el método automático fundado en el estudio de figuras de suelos de pavimentos puede apresurar el desarrollo y disminuir la subjetividad del desenlace. (INGE CUC, 2012).

La Universidad de Texas, en Austin, propuso un algoritmo de tratamiento de figuras para revelar rajaduras en pavimentos, la cual se basa fundamentalmente en el detalle de la figura, para igualar el destello y el contraste de la figura; partición de la figura en cuadros de 8 x 8 píxeles, para encontrar las rajaduras comparando su contraste y su distribución, su alejamiento y su rumbo y traslado de secciones acopladas que están por debajo del umbral preestablecido. (B. Xu and Y. Huang, 2015).

Lo principal es arreglar algunas particularidades que beneficien a las posteriores ejecuciones sobre la figura. En la pista el brillo, la oscilación y la temperatura producen denotación en la adquisición de la figura, puesto que los elementos pueden estar atrapados a anomalías que deben ser tratadas. Los procedimientos originados para examinar y sacar las particularidades se fundamentan en la división, que es la separación de la figura. Esta división es de vital importancia ya que esta sección es donde se agrupa la indagación importante, pero el dividir la figura puede ser una actividad difícil ya que en reiteradas ocasiones los métodos simples como la división bimodal no daban soluciones buenas y es imprescindible enormes deducciones y recursos computacionales para efectuar la actividad. (Z. Wan-zhi, 2013, 2013).

Hay otros métodos que se han desarrollado para alcanzar la división de figuras como los algoritmos de máxima entropía, la técnica otsu, el crecimiento por regiones, la técnica de histograma de vecinos, que son los que prometen una deseable solución para dividir en diferentes escenarios, cuya ocupación primordial es incrementar los pixeles repetitivamente empleando similitudes entre píxeles con un tamaño semejante. Estos algoritmos poseen una enorme porción de inconvenientes cuando los llevamos a las pistas, pierden su efectividad. Ya que, en los ambientes fuertes, el brillo constante y la oscilación, permiten que estos algoritmos pierdan su capacidad ya que son muy susceptible al ruido, necesita de una señal de inicialización para producir un buen o mal resultado de segmentación

y cuando son repetitivos es cuando consumen demasiado recurso computacional (J. Zhu, 2015).

1.2. Trabajos previos

En la investigación “Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks” realizado por (Young-Jin Cha & Wooram Choi,) en donde se describe una de técnica de procesamiento de imágenes. Las IPT’s se han implementado para detectar defectos de infraestructura ya que se utilizan principalmente para manipular imágenes, para extraer características defectuosas, como grietas en superficies de hormigón y acero. Sin embargo, las situaciones del mundo real que varían ampliamente (cambios de luces y sombras) pueden generar desafíos para la adopción generalizada de IPTs. Para superar estos desafíos, este artículo propone un método basado en la visión que utiliza una arquitectura profunda de redes neuronales convolucionales (CNN) para detectar grietas de concreto sin calcular las características del defecto. Como las CNN son capaces de aprender características de imagen automáticamente, el método propuesto funciona sin la conjugación de IPT para extraer características.

En la investigación “Automatic Identification of Cracks in Road Surfaces” realizado por (Pynn, Wright y Lodge, 2015) en donde especifica la exigencia de un boceto y el despliegue de un método de escaneo de grietas. Se nombró HARRIS. Las figuras son recogidas por el carro de comprobación y su desarrollo se realiza en dos etapas. En el primer escenario se requiere quitar y disminuir las figuras, y en la etapa final se ejecuta el estudio y la estimación de los defectos de modo off-line. Esta investigación sobresale por la manipular tres cámaras que monitorean el extenso tamaño de la pista, así mismo las unen y las estructuran en el post procesamiento para producir una figura acabada y dispuesta para ser estudiada. En el estudio de las figuras se efectúa equiparando el histograma de una base de datos libre y donde los datos recogidos puedan calcular proporción de errores sobre la pista. Los resultados obtenidos alcanzan una puntuación de 75%. En este método, las imágenes recolectadas muestran detallada información ya que se utiliza tres cámaras para el reconocimiento de grietas en el asfalto. Estos resultados demuestran la relación entre la profundidad de la textura y la fricción de alta velocidad.

Según (Chang'an de Xi'an, 2015) en su artículo "Beamlet Transform Based Pavement Image Crack Detection"; manifiesta la aparición de rajaduras a través figuras del área de la pista, empleando el método de esta investigación con la finalidad de reconocer rayas, circunferencias u elementos en figuras con demasiado ruido. La exploración admitió verificar que el empleo de la transformada Beamlet para localizar rajaduras es invulnerable al ruido mostrado en las figuras, lo que asegura eficiencia en el estudio al momento de realizar el procesamiento. La propuesta puede detectar y clasificar las grietas en el pavimento en un 85%. Los resultados demuestran que este método no se puede usar para detectar los defectos con un área grande. Además, se necesitan pruebas más exhaustivas para que el algoritmo sea más práctico para detectar grietas.

Según (Yong HU, 2014) en su artículo "Automatic Pavement Crack Detection Using Texture and Shape Descriptors"; esta investigación sugiere un fresco enfoque de localización de rajaduras en el asfalto de manera instantánea originada en el estudio de trama y detalle de formas. Equiparando con localizador de bordes habituales, las soluciones efectivas enseñaron que todas las rajaduras son reveladas satisfactoriamente por la técnica sugerida, aun con base de trama difícil o con alumbrado disparejo. Los resultados obtenidos alcanzan un porcentaje del 70% donde se demuestran que las imágenes muestran una grieta transversal en la superficie del pavimento con iluminación desigual. En algunas regiones tienen un alto brillo, fuerte fondo de textura. La imagen es discontinua en su estructura. Además, muestra una ligera grieta longitudinal en la superficie. Los resultados demuestran que la textura compleja, iluminación desigual y no uniforme. Con estos antecedentes es un gran desafío desarrollar este sistema de detección y evaluación de dificultades en el asfalto.

En la investigación "Recognition of pavement surface crack" realizado por (Zhang y Li, 2015) en donde sugiere una construcción hardware para un método de verificación de rajaduras de figuras en las pistas y explica cada uno de sus convenientes submétodos. Por lo tanto, estudia y equipara procedimientos de localización de límites y márgenes como los algoritmos de Sobel o Prewitt y sugiere un moderno algoritmo arreglando el rendimiento. Este método propuesto destaca y saca beneficio de otros componentes tecnológicos igualmente del estudio de

figuras para ejecutar con sus objetivos, como por ejemplo la utilización de instrumentos GPS y de alumbrado. La propuesta verifica rajaduras considerables a 3 mm con una exactitud del 90%. Los resultados obtenidos demuestran que la utilización de instrumentos de GPS y de alumbrado se obtienen mejores resultados.

Según (Henrique Oliveira y Paulo Lobato Correia, 2016) en su artículo “Automatic Road Crack Segmentation Using Entropy And image Dynamic Thresholding”; da a conocer una nueva opción para la identificación de rajaduras y la distribución automática de acuerdo con las figuras poseídas de gran rapidez. Las figuras que quedan son pre procesadas empleando filtros morfológicos para disminuir la alteración de magnitud de pixeles. El umbral dinámico se coloca para detallar los pixeles opacos en las figuras, y donde estos afectan a los pixeles que contienen rajaduras. Este método distribución marca la figura de acuerdo a lo hallado: rajaduras horizontales, verticales, muchas o nada. El método propuesto alcanza un 80%. Para el módulo de preprocesamiento, utilizó un elemento estructurante en forma de disco con un radio de 5 píxeles. Los resultados obtenidos demuestran que la presente propuesta puede manejar imágenes utilizando distintos sensores de imágenes, que muestran una robustez interesante, que se refleja en los valores de recuperación similares.

Según (Ma, Wang, Zha, Di y Z hu, 2014) en su artículo “Pavement Cracks Detection Based on FDWT”; Manifiesta una nueva opción en la localización de defectos en asfaltos por medio de la fracción diferencial y transformada de wavelets. El empleo de la fracción diferencial permite arreglar los indicadores de media y de gran frecuencia Posteriormente se emplea la FDWT para filtrar la distorsión que enseña unida a la umbralización acondicionada para la división y localización de lugares en donde existan rajaduras. El método propuesto alcanza el 70%, donde las imágenes de pavimento con altos niveles de textura superficial que dificulta la detección de grietas. Los resultados obtenidos de este método son una buena propuesta para la identificación de grietas.

Según (Conferencia Internacional en Procesamiento de Imágenes,2014) en su artículo “Automation of Pavement Surface Crack Detection Using the Continuous Wavelet Transform”; Hace referencia a una nueva alternativa en la automatización en donde localiza rajaduras en figuras del área del asfalto. Esta técnica se

manifiesta en la Wavelet Transform. Y dónde saca la información del patrón y de la esquina de la rajadura, seguidamente se investigan los valores altos de los factores de Wavelets y donde son estudiados. En la parte final se manifiesta el post tratamiento obteniendo como resultado una figura binaria la cual muestra la existencia o no de rajaduras en las figuras del área del asfalto. El método propuesto ha sido aplicado a las imágenes de carreteras alcanzando el 75%. Las grietas se detectan bien con más o menos ruido según la textura. Por lo tanto, en imágenes de laboratorio se detecta la grieta, pero hay algo de ruido debido a una textura fuerte. Los resultados demuestran que las imágenes basado en wavelet tiene una representación compleja, donde la información del ángulo permite considerar sólo coeficientes que pertenecen a la grieta.

Según (Beijin, 2020) en su artículo “Determination of the Varieties and Characteristics of”; el artículo hace referencia a la descripción de una técnica de procesamiento de imágenes donde aparecen defectos en las semillas, por lo tanto se emplean para detección de características defectuosas en las semillas forestales, por lo que utilizaría SVM. El método propuesto alcanza el 85%, que es una buena propuesta para la identificación de grietas.

Según (Coelho, 2018) en su artículo “Mussel Classifier System Based on Morphological Characteristics.”; en este escenario manifiesta el escaneo seguidamente del conteo y la clasificación de mejillones en cultivos marinos para la producción de semillas. En esta aplicación se emplean una cámara que monitorea el extenso cultivo marino para que luego pueda ser estudiada. En este método las figuras capturadas manifiestan una detallada información en relación a la profundidad de la textura. Los resultados obtenidos alcanzan el 82%.

Según (Gunaseelan JayaBrindha, E. S. Gopi Subbu, 2017) en su artículo “Ant Colony Technique for Optimizing the Order of Cascaded SVM Classifier for Sunflower Seed Classification.”; Este es un proceso que depende del punto de inicialización y del tiempo para que los elementos a identificar puedan ser tratados, no obstante; existe la posibilidad de que se produzca un error humano al identificar la ODV. Por lo tanto, existe la necesidad de que la técnica de visión artificial automatice el proceso de identificación de ODV en el laboratorio de análisis de semillas. Este método propuesto alcanza el 80%.

Según (Ruojing Wang; Eric Neufeld, 2018) en su artículo “Identification of Morphologically Similar Seeds Using Multi-kernel Learning”; esta investigación nos comenta el uso de análisis de imágenes digitales para la identificación de semillas con la finalidad de reconocer elementos en imágenes con demasiado ruido. El análisis de imágenes para la identificación de semillas se ha estudiado previamente y se han logrado buenas tasas de reconocimiento. La propuesta puede detectar y clasificar figuras en un 72%.

Según (Carlos, Percy 2016) en su artículo “Procesamiento digital de imágenes: Esta investigación manifiesta la determinación de un fresco tamaño de imágenes durante el crecimiento de semillas pequeñas de manera instantánea”; esta técnica muestra el uso de figuras digitales para medir la longitud ecuatorial de las frutas, sin embargo, en algunas regiones cuenta con un fondo bastante fuerte. Esto quiere decir que las figuras son discontinuas y no uniformes. Este método propuesto alcanza el 72%.

Según (Angel, Roger 2017) en su artículo “Diseño e implementación de un sistema de reconocimiento y manipulación de frutas utilizando visión artificial y brazo robótico industrial”. Esta propuesta nos da a conocer una nueva opción en el reconocimiento y manipulación en el campo de la inteligencia artificial, utilizando este elemento las figuras muestran una robustez interesante. Este método propuesto alcanza el 85%.

Según (Tafur, Sobrado 2018) en su artículo “Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot”. El empleo del brazo robótico hace referencia a la automatización, técnica que permite localizar con más exactitud el área a tratar y dónde como resultados se obtienen figuras con rapidez y altos niveles de textura. Este método propuesto alcanza el 82%.

1.3. Teorías relacionadas al tema

1.3.1. Pistas asfaltadas

"Las pistas asfaltadas forman parte un armazón de múltiples fases edificadas encima del sostén de la vía para aguantar y dispersar trabajos realizados por los carros y renovar la situación y el bienestar para los mismos" (Manual de carreteras Mantenimiento Vial, 2014). Así mismo, se detalla de la siguiente manera:

Capa de revestimiento: Se manifiesta en la parte alta del asfalto, que puede de concreto de cemento.

Cimiento: Se da en la parte baja de la capa de revestimiento, que sujeta, comparte y cede las cargas ocasionadas por los vehículos.

Subcimiento: Es un manto de material con una densidad de diseño que sujeta al cimiento.

Figura 1.Proceso de pistas de asfalto.

Fuente: (Construcción de asfalto, 2014)

Subcimiento: Es un manto de material con una densidad de diseño que sujeta al cimiento.

Modelo de asfalto

a) Asfalto blando

“Este armazón está combinada por niveles de cimiento y subcimiento. Así como niveles de revestimiento con componentes y aditivos” (Manual de carreteras Mantenimiento Vial, 2014). Así mismo, la capa de revestimiento manifiesta un tratamiento superficial.

Figura 2.Soporte asfáltica. Fuente: (Contingencia vial, 2014)

b) Asfalto semirrígido

“La distribución del asfalto se encuentra conformada por capas con un grosor bien definido” (Manual de carreteras Mantenimiento Vial, 2014). Además, se considera como asfalto semirrígido la distribución compuesta materiales de calidad.

c) Asfalto rígido

“La dispersión en esta fase se encuentra conformada directamente por una capa de subcimiento granular, así mismo, este puede ser controlada con cal o cemento además de un manto de asfalto aglomerante más aditivos”. (Manual de carreteras Mantenimiento Vial, 2014) Dentro de asfaltos rígidos podemos encontrar tres etapas:

Figura 3.Estructura rígida del asfalto. Fuente: (Carreteras viales, 2014)

1.3.2. Clasificación no Supervisada

"Las pistas con grietas en el asfalto (<0.03 mm) se nombran como fisuras, en esta investigación da a conocer los daños que se ven en un asfalto"(Manual de carreteras Mantenimiento Vial, 2014).

Grietas en bordes

"Ocasiona una rajadura de forma de silueta en el asfalto; tratando de extenderse a través de las uniones." (Manual de carreteras Mantenimiento Vial, 2014).

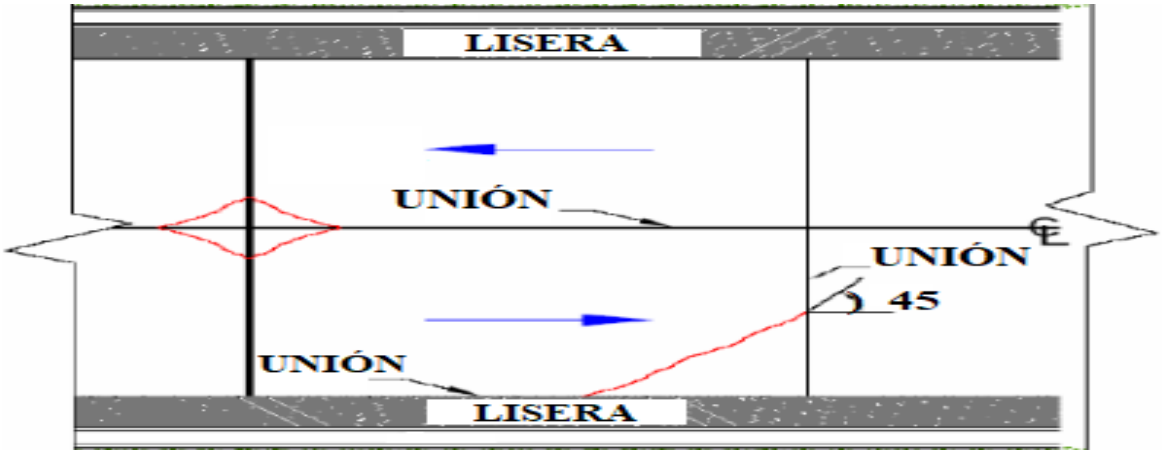


Figura 4.Propiedades de grietas. Fuente: (Estructura pavimental,2014)

Figura 5.Grieta longitudinal.Fuente (Construcción de asfalto, 2014)

Grietas transversales

“Las grietas transversales alcanzan una extensión de daño considerable en el área afectada de la pista.” (Manual de carreteras Mantenimiento Vial, 2014)

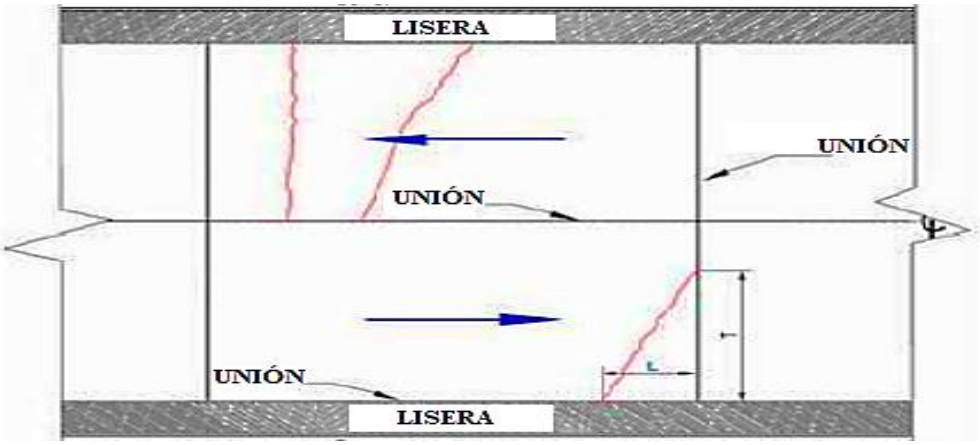


Figura 6.Distribución de la grieta.

Fuente (Estructura pavimental, 2014)

Figura 7.Grieta de pista con asfalto.

Fuente (Construcción de asfalto, 2014).

1.3.3. Métodos ó técnicas para identificar grietas

En el Perú, los modelos de evaluación manuales más utilizados son: la inspección de pavimentos de la Superintendencia de Transporte Terrestre (SUTRAN), el indicativo de Circunstancia de Pavimentos y el Programa de Investigación Estratégico de Carreteras (SHRP) (Revista CUC, 2016). Sin embargo, ya se están integrando tecnologías para contribuir a la gestión y a la intervención de carreteras; además, en el ámbito mundial, ya que han ido desarrollándose

tecnologías para la localización y el estudio de información de forma automática o semiautomática. (M. S. Townes, 2014). La inspección visual de pavimentos consiste contratar personas para que llenen los formatos a mano. Este procedimiento actúa de manera pausada, peligrosa y de generan elevadas pérdidas de dinero al momento de contratar personal. Un componente que acceda a disminuir el grado de variabilidad da origen a la estandarización la sistematización basado en el análisis de tratamiento de imágenes. (INGE CUC, 2012).

La Universidad de Texas, en Austin, propuso un algoritmo de tratamiento de figuras para revelar rajaduras en pavimentos, la cual se basa fundamentalmente en el detalle de la figura, para igualar el destello y el contraste de la figura; partición de la figura en cuadros de 8 x 8 píxeles, para encontrar las rajaduras comparando su contraste y su distribución, su alejamiento y su rumbo y traslado de secciones acopladas que están por debajo del umbral preestablecido. (B. Xu and Y. Huang, 2015).

Lo principal es romper paradigmas que ayuden al porvenir de las operaciones sobre los retratos. En el asfalto predomina el ruido tales como: el destello, oscilación y la temperatura. Esto se manifiesta en la adquisición de un retrato, de modo qué los objetos están sujetos a irregularidades listas para ser tratadas. Los métodos para estudiar y sacar las cualidades se originan en la división de figuras, que es la separación del retrato. La segmentación permite seleccionar la parte de interés, pero al momento de desintegrar un retrato la tarea suele ser muy complicada ya que años atrás no generaban resultados positivos y era inevitable deducir recursos conmensurables para concretar dicha actividad. (Z. Wan-zhi, 2013).

Las técnicas conmensurables permiten la evolución de los algoritmos de segmentación que favorezca la examinación para su entendimiento en la técnica, ya que de poder realizar algoritmos más rápidos es aceptable aumentar el detalle y bajar la dificultad conmensurable. La presente investigación, pretende proporcionar algoritmos para desarrollar nuevas técnicas que trascienda en las técnicas de la segmentación. (JLB Bonilla, 2018). Podemos encontrar métodos actuales que se han creado dentro de la investigación para poder lograr la segmentación como los algoritmos de máxima entropía, el método otsu, el

crecimiento por regiones ya que generan óptimas soluciones para segmentar en varias etapas, cuya competencia primordial es amplificar los pixeles repetitivamente empleando verificaciones entre pixeles con una mesura de semejanza. Dado a la capacidad de estos algoritmos pierden su efectividad al momento de dirigirlos y probarlos en las pistas. Esto se manifiesta en ambientes fuertes, alumbrado constante y la oscilación donde los algoritmos manifiestan inconvenientes porque son perceptivos al ruido, necesita un bue punto de partida para ocasionar un favorable o una incorrecta segmentación y como son repetitivos procuran gastar demasiados recursos conmensurables. (J. Zhu, 2015).

1.3.4. Imágenes

"Una imagen está compuesta por dos dimensiones f (x,y). Estas coordenadas se muestran en un plano donde la amplitud es la intensidad (x,y). Sin embargo, se habla de una imagen digital cuando los ejes como también los cálculos de magnitud (f) son potentes y finitos" (Scarel, 2010).

La imagen digital se encuentra mesurada por un número finito de componentes donde cada uno tiene un valor particular. Estos componentes son denominados píxeles, este último expresa la unidad mínima de la imagen en mención.

En la Figura 8, podemos observar un retrato con 256 niveles de intensidad. Así mismo, estos pixeles representan el grado de intensidad de luz en gama de grises.

Expandir el retrato en cualquiera del mismo, se estiman estos valores de la siguiente manera: matriz Nij con los ejes x=i , y=j.

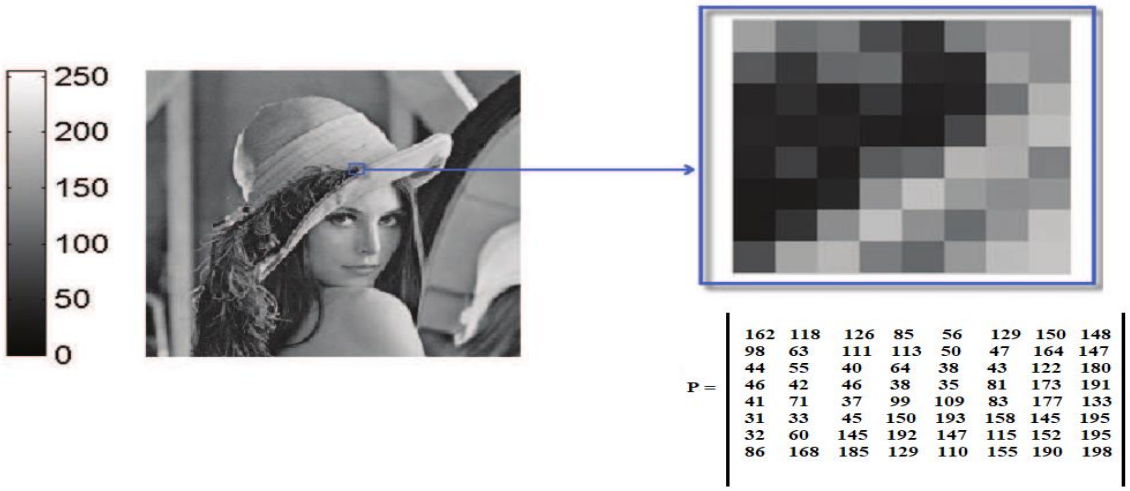


Figura 8.Retrato en gama de grises. Fuente *(Imagen digital, 2011)*

Un factor bastante valioso en un retrato digital es la resolución. Esta denota una ración de pixeles que contiene un retrato. Es utilizada para catalogar los mecanismos con los retratos digitales, estos pueden ser computadoras, impresoras, escáneres, etc. La resolución en su máxima expresión manifiesta una cantidad de píxeles que componen un retrato de bits. La calidad de un retrato va a depender concretamente de la resolución.Un buen modelo de clasificación de imágenes debe ser invariable al producto cruzado de todas estas variaciones, al mismo tiempo que debe conservar la sensibilidad a las variaciones entre clases (Karpathy 2016).

Tipos de Figuras

a) Figuras en color

El motivo para calificar un retrato digital en tono se manifiesta de la siguiente manera:

Por ejemplo, un campo de color rbg (ocasionalmente el más utilizado), muestra a cada uno de los pixeles como un color inventando a partir de algunas porciones como lo son: el rojo, verde y el azúl. Esta muestra, genera una matriz de varios niveles de intensidad (3 niveles) que pertenecen a la intensidad del color (rojo, verde, azul).

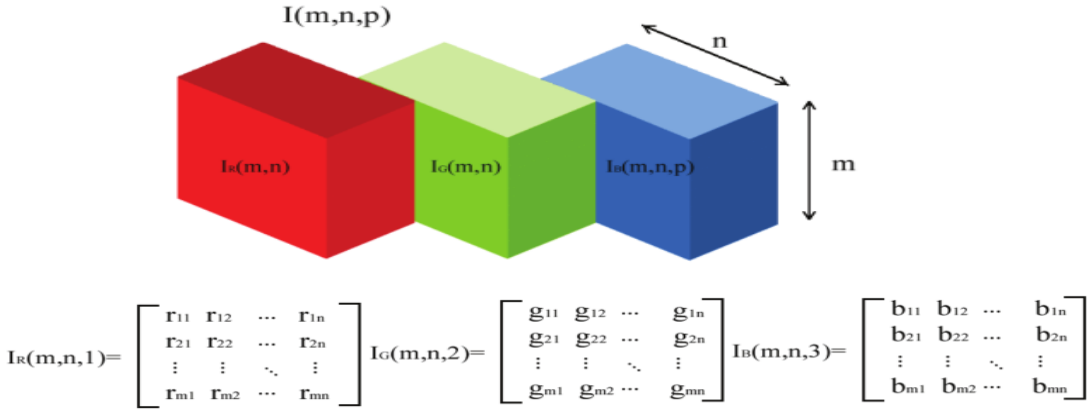


Figura 9.Plano de color RBG. Fuente: (segmentación de figuras, 2011)

Para laborar con matrices para explicar imágenes usualmente se da en la visión computacional, no obstante, no es la única alternativa para mostrar un retrato en color. Sin embrago, cuando se maneja el procesamiento de figuras en FPGA, se olvida la idea de matriz para ceder pase a la noción de flujo de información por bus.

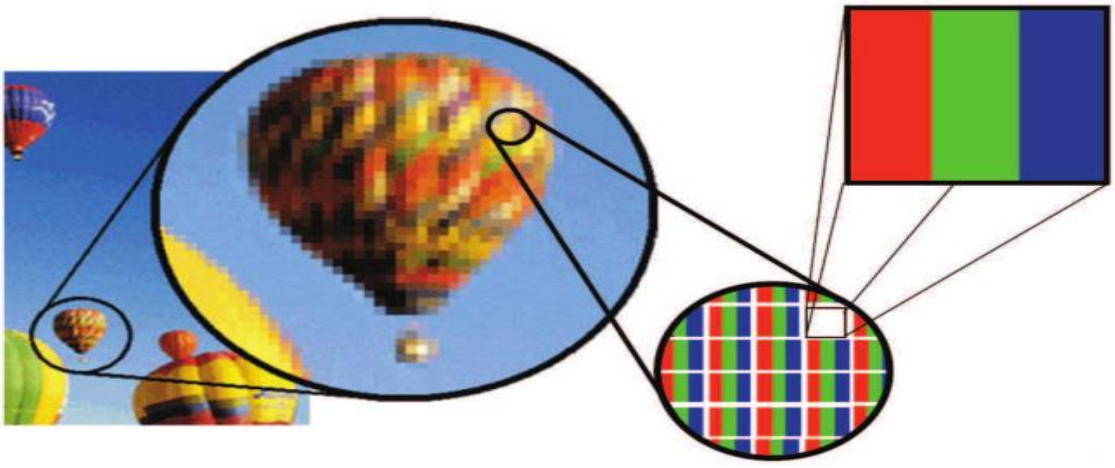


Figura 10.Retrato en color. *Fuente (figura bidimensional, 2011)*

En un retrato de rbg cada píxel se encuentra mesurado por una importancia de intensidad (valor) que afecta a cada elemento primario. Como el resultado el pixel se aproximaba dada la "porción" de magnitud que contenga dicho elemento.

Así que, el color blanco será compuesto con el mayor grado de intensidad para sus 3 componentes. De manera contraria con el color negro porque pasar a disminuir la intensidad de sus componentes.

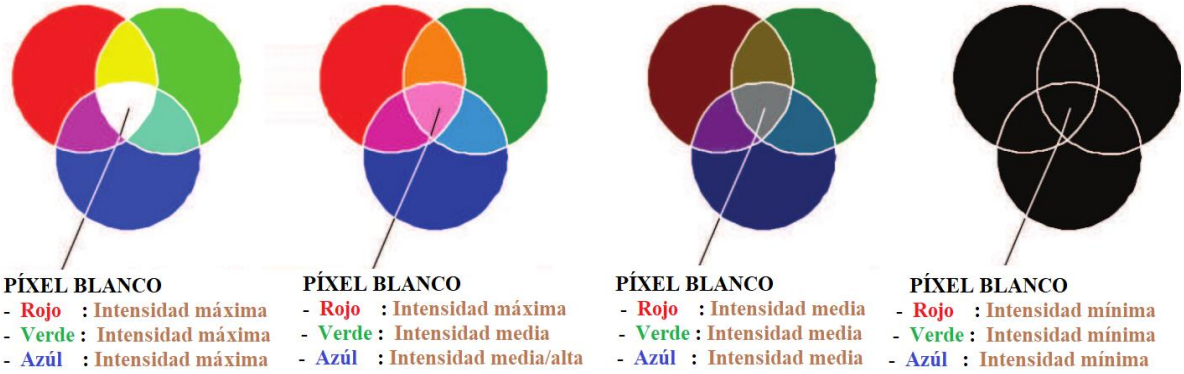


Figura 11.Píxel y sus componentes. Fuente: (figura bidimensional 2011)

b) Variables del color

Para un mejor entendimiento del por qué hay distintos ámbitos, se empezará hacer una pequeña explicación de las propiedades del color: matiz, luminosidad, tono y la saturación.

Matiz. Denota la importancia pura de un color, donde está ubicado el espectro, precisa de una onda dominante y del atributo que permite llamar a los colores: azul, naranja, verde, etc

Luminosidad. Se obtiene del resultado de mezclar colores como: blanco o negro y guarda semejanza con la matriz. Además, manifiesta la porción del color, ya sea el caso.

Saturación. Denota la importancia del color con conexión al gris. Los colores menos saturados se presentan grisáceos con un grado superior de impurezas y con menor intensidad de luz.

Figura 12.Saturación RGB. Fuente (segmentación de figuras, 2011)

c) Espacios del color

En este apartado los ambientes de tonos son un instrumento de vital importancia en el tratamiento digital de figuras, ya que generan el estudio de cada uno de los píxeles desde otra perspectiva. De esta manera se poder conseguir toda la indagación detallada que se encuentra en la figura. Las labores más frescas efectuadas en este escenario se agrupan con la distribución de figuras en tonos, verificación de elementos, estudio de trama, configuración matemática, normalizar figuras en tonos, etc. Las técnicas no lineales son repetitivas en los ambientes de tonos, ya que exploran para hacer algunas características de la figura. Podemos encontrar varios ambientes de tono considerando las carencias desiguales que van de la mano de la doctrina del sentido humano, incluso la técnica de tono disminuido empleado en el estampado encima del papel. En este escenario se manifiestan los más usados en la percepción del computador y en tratamiento de figuras. Se resalta que ciertos de estos ambientes de tonos no poseen como objetivo realizar la observación de tonos lo más cerca posible a la realidad, sino que son inexactas matemáticas, por lo general no lineales, que originan el procesamiento de algunas pertenencias de las figuras.

d) El RGB

Son unos de los más empleados por los métodos sistematizados para procesar y crear tonos en pantallas. Se origina en el gesto de "resumen aditivo" donde el entusiasmo de la iluminación correspondiente al rojo, verde y azul son añadidas entre sí para alcanzar variedad de tonos insertando el negro y el blanco.

La exhibición del esquema del tipo RGB se emplea a través de una cubeta indivisible con ejes R, G, B. La partida (0,0,0) simboliza el negro y los ejes (1,1,1) el blanco. La cúspide de la cubeta en cada plano R, G, B de ejes (1,0,0), (0,1,0) y (0,0,1) simbolizan los tonos principales rojo, verde, y azul. Los vértices que quedan (1,0,1), (0,1,1) y (1,1,0) son tonos provisionales y adicionales al verde, rojo y azul.

El modelo rbg manifiesta una buena importancia en el procesamiento de imágenes. Tal es el caso el de comparar colores.

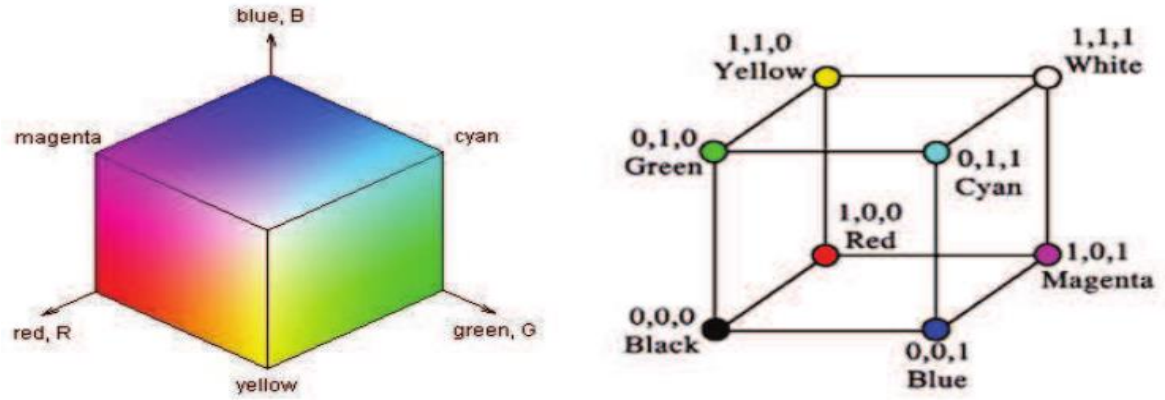


Figura 13.Modelo de color RBG. Fuente: *(figura bidimensional, 2011)*

1.3.5. Procesamiento de imágenes

Para esta investigación describimos cada concepto que son necesarias para la identificación de grietas en el asfalto a través de tres fases:

a. Fase de procesamiento

En esta primera fase buscó perfeccionar las cualidades de las figuras de entrada, por lo que pueden presentar características que no son apropiadas para el estudio del análisis, debido a este inconveniente y a los factores externos en el preciso instante de la captura. Esto se va a realizar con el objetivo de asegurar que el desarrollo de detección se manifieste de manera óptima.

Cambio a gama de grises

Están compuestas por beneficios que manifiestan exclusivamente la magnitud de los pixeles. Ya sea únicamente tonalidades grises abarcando a partir del color negro hasta llegar al color blanco. Sin embargo, para poder medirlo se utiliza la cantidad de 8 bits, lo que demuestran que hay 256 matices de grises,

abarcando desde el "0" que implica la pérdida global de brillo y por otra parte el "255" la disposición total.

El cambio de una figura rbg a una figura a escala de grises se manifiesta con la suma de sus elementos de pixeles a color y se manifiesta de la siguiente manera:

$$I = 0.2989 \,.\, R + 0.5870 \,.\, G + 0.1140 \,.\, B$$

Aquí R es la estimación del elemento rojo, G la estimación del elemento verde y B la estimación del elemento azúl. (Solomon, 2011).

Transformación logarítmica

Es un método que sirve para perfeccionar el contraste de una figura. También se basa en corregir el rango de la intensidad de cada píxel, como se muestra a detalle. (Solomon, 2011)

$$I_0 = \ln(I_i)$$

En este escenario I_i, es la matriz de potencia de la figura de acceso e I_0 el molde de la figura de salida. Es importante que la clase de la figura en escala de grises empieza 0 a 25, es probable que el registro sea indifinible aunque I_i=0. En la actualidad se tiene la siguiente expresión. (Solomon, 2011).

$$I_0 = C \,.\, ln\,(1 + (e^{\,\sigma} - 1) I_i)$$

Aquí σ es un dígito racional verdadero, que simboliza el factor de serie que domina la calidad de acceso y C escala la estimación de escape dentro de lo permitido por la figura (0-255). (Solomon, 2011).

$$C = \frac{255}{\log(1 + \max(I_i))}$$

Variación logarítmica

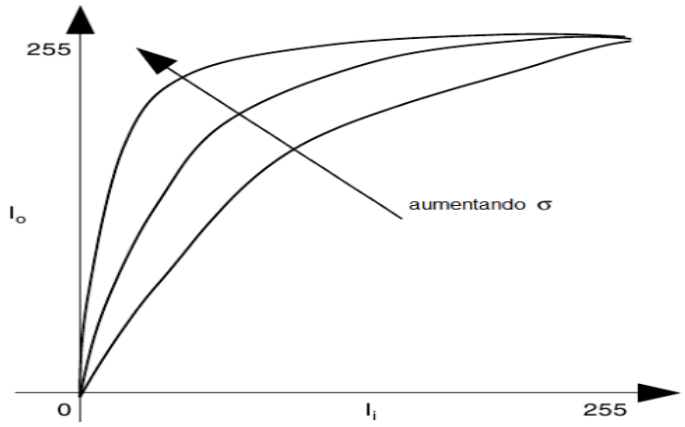


Figura 14.Transformación logarítmica.

Fuente: *(figura bidimensional, 2011)*

En la imagen podemos observar qué cuánto más alto es el valor de σ, más enorme será la jerarquía dinámica de los territorios opacos de la figura, disminuyendo la jeraquía dinámica de las regiones despejadas.

Filtro de color

Encontrar el tono preciso en la figura, es un método trabajoso ya que la misma condición se pueden hallar tonos de distintos colores acorde a sus propios valores ya sea rojo, verde, azúl. Esto demuestra que ubicar el rango rbg sea un poco complicado.

Para resolver este inconveniente se puede deducir que el área rbg como un cubo es el color que necesita detectar, el cual se llamará punto de interés.

Para resolver si el píxel en una figura afecta a un tono determinado se calcula el espacio con el punto de interés. Éste se da de la siguiente manera:

$$PI(R,G,B) = \{r_i, g_i, b_i\}$$
$$PE(R,G,B) = \{r_e, g_e, b_e\}$$

Los elementos r,g y b conforman una categoría de 0 a 255. El espacio euclídea se manifiesta de la siguiente manera:

$$dE = \sqrt{(PI(R) - PE(R))^2 + (PI(G) - PE(G))^2 + PI(B) - PE(B))^2}$$

b. Fase de detección

El principal objetivo de esta fase es la identificación del área a tratar. Para poder entenderlo, será necesario analizar conceptos:

Thresholding

En el proceso de figuras digitales, el thresholding es el procedimiento más fácil para segmentar figuras. Al comienzo de la figura en escala de grises, podemos utilizar el thresholding para establecer figuras binarias.

Los procesos de umbral más fáciles sustituyen cada píxel en una figura con un píxel negro si la magnitud de la figura {\ displaystyle I_ {i, j}} I _ {{i, j}} es menor que cualquier constante T (es decir, {\ displaystyle I_ {i, j} <T} I _ {{i, j}} <T), o un píxel blanco si la magnitud de la figura es mayor que esa constante.

En la figura de modelo, se obtiene como resultado que el arbusto se convierta totalmente negro y la nevisca blanca se convierta totalmente blanca.

Imagen original Thresholding utilizado en una imagen

Figura 15.Segmentación de thresholding a escala de grises.

Fuente: (procesando imágenes,2014)

Thresholding

Las figuras coloridas también se pueden rastrillar. Una perspectiva es elegir un umbral desprendido para cada uno de los integrantes RGB de la figura y posteriormente unirlos con un procedimiento AND. Da a conocer la estructura en que actúa la cámara y cómo se acumulan los apuntes en el computador, pero no afecta a la figura en que los individuos examinan el color. Por consiguiente, los prototipos de color HSL y HSV se utilizan con mayor reiteración; hay que tener en

cuenta que, dado que la tonalidad es una porción radial, necesita un umbral radial. También es aceptable utilizar el prototipo de color CMYK.

Thresholding Automático

Es una magnífica forma de sacar información valiosa cifrada en píxeles entretanto se disminuye el ruido de fondo. Esto se puede obtener empleando un bucle de retroalimentación para lograr tener el valor del umbral antes de cambiar la figura única en escala de grises a binaria. El concepto es alejar la figura en dos fracciones; el fondo y el primer plano.

c. Fase de medición

Esta fase permitirá abordar las mediciones que sean pertinentes para resolver el tipo de daño y el nivel de rigidez. Sin embargo, es importante dar a conocer lo siguiente:

Medida de la longitud de un objeto

Para obtener la medida de la extensión de un objeto situada en la figura binaria es crucial la intervención morfológica.

El enflaquecimiento de un objeto de silueta prolongada con anchura firme genera una porción de marcas que establece una raya que se inicia a medida a las orillas del elemento. Teniendo en cuenta la zona que llena un píxel en el área existente, se puede llegar a precisar el acercamiento de la distancia de un elemento refiriéndose al número de píxeles que integran su centro, sin embargo, se debe multiplicar este último valor por el tamaño de la longitud en milímetros.

Los objetos que no manifiestan un enflaquecimiento ni mucho menos una silueta alargada puede causar derrames en el eje, esto provocaría un error en la medida, ya que habría más píxeles de los que se necesiten. Por ello, es fundamental exterminar la salida de la estructura, para ubicar los puntos de terminación y de derrame.

Los puntos de terminación se reconocen como píxeles con estimación "1", que poseen únicamente un vecino de "1er" nivel, sin embargo, los puntos de

derrame se reconocen como píxeles con terminación "1" que poseen 3 o 4 vecinos juntos.

Medida de la anchura de un objeto

Ofrece hacer la medida de la anchura media y del extenso de un elemento prolongado en una figura binaria.

Dicho lo anteriormente se puede concretar pasando la extensión absoluta del elemento y calcular la porción de píxeles de terminación "1".

Cuando se carga con la figura del objeto, se comprenden los puntos extremos de comienzo y fin.

d. Fase de clasificación

Ofrece hacer la medida de la anchura media y del extenso de un elemento prolongado en una figura binaria.

Se investigan 3 modelos de grietas: longitudinal, transversal y de rincón:

La grieta longitudinal

La unión alta y baja del asfalto.

La unión alta e izquierda o alta y derecha, que regularmente el tamaño del daño sea superior al promedio del largo del asfalto.

La unión baja e izquierda o baja y derecha, que regularmente el tamaño del daño sea superior al promedio del largo del asfalto.

La Grieta transversal

La unión del extremo derecho e izquierdo del asfalto.

La unión del extremo derecho y alta o derecha y baja, que regularmente el tamaño del daño sea superior al promedio del ancho del asfalto.

La unión del extremo izquierdo y alto o izquierdo y bajo, que regularmente el tamaño del daño sea superior al promedio del ancho del asfalto.

La grieta del rincón

La unión alta e izquierda o alta y derecha, que regularmente el tamaño del daño no pase el promedio del largo del asfalto y sea superior.

1.3.6. Clasificadores

Red Neuronal Convolucional (CNN)

Las personas pueden afiliar figuras en escala de grises con tonos en la misma, pero brinda a las personas un obstáculo para ocupar los tonos. La tonalidad manual solicita mucho trabajo. No obstante, CNN puede intuir particularidades velozmente. En la actualidad, las redes neuronales convolucionales se han transformado en el dominador tecnológico para labores relacionadas con la figura.

En enseñanza profunda, una red neuronal convolucional (CNN) es una categoría de redes neuronales profundas, que se adapta con mayor continuidad al estudio de figuras visuales.

Poseen aplicaciones en la comprobación de figura y video, método de verificación, distribución de figuras, estudio de figuras médicas, lenguaje natural, procesamiento y finanzas.

Las redes convolucionales se basaron en los desarrollos biológicos en que el modelo de conectividad entre las neuronas se parece a la estructura del revestimiento ocular animal. Las neuronas corticales individuales contestan a los impulsos solo en una región restringida del terreno ocular conocida como terreno receptivo. Los terrenos receptivos de distintas neuronas se incorporan parcialmente de modo que ocupan todo el terreno ocular.

Las CNN emplean parcialmente escaso procesamiento anticipado en semejanza con otros algoritmos de distribución de figuras. Esto implica que la red asimila los filtros que en los algoritmos habituales fueron diseñados a mano. Esta libertad del conocimiento anticipado y el esfuerzo del hombre en el diseño de particularidades es una gran ventaja.

a. Aquitectura

Las redes convolucionales radican en una capa de acceso y una de escape, así como múltiples capas escondidas. Estas capas escondidas de una CNN generalmente se basan en una sucesión de capas convolucionales que se enlazan

con un acrecentamiento. La ocupación de activación es usualmente una capa y luego es continuada por convoluciones complementarias tales como capas de concentración, capas totalmente unidas y capas de normalización, llamadas capas escondidas porque sus accesos y escapes están disfrazadas por la ocupación de agilidad y la convolución de conclusión.

Aunque las capas se comprenden coloquialmente como convoluciones, esto es solo por pacto. Matemáticamente, es técnicamente un resultado de correlación cruzada. Esto tiene consideración para las listas en la matriz, ya que afecta cómo se determina el peso en un punto de índice específico.

b. Convolucional

Al sistematizar un CNN, el acceso es un resorte con aspecto de número de figuras: pico de figura, amplio de figura, depresión de figura. Después de transitar por una capa convolucional, la figura se aísla en un plano de particularidades: cantidad de figuras, pico del plano de particularidades, amplio del plano de particularidades, canales del plano de particularidades. Una capa convolucional dentro de una red neuronal debe tener las siguientes cualidades:

Núcleos convolucionales determinados por un amplio y pico (hiperparámetros).

La cantidad de canales de acceso y canales de escape (hiperparámetros).

El fondo del filtro de convolución (los canales de acceso) debe ser idéntico al número de canales (profundidad) del plano de particularidades de acceso.

Las capas convolucionales convolucionan el acceso y llevan su solución a la posterior capa. Esto es semejante a la solución de una neurona en el revestimiento ocular a un impulso específico. Cada neurona convolucional encausa datos solo para su terreno pertinente. Aunque las redes neuronales de alimentación directa completamente enlazadas se pueden utilizar para estudiar particularidades y clasificar datos, no es práctico adaptar esta arquitectura a las figuras. Sería imprescindible una gran porción de neuronas, incluso en una arquitectura poco recóndita, debido a las dimensiones de acceso muy colosales asociados con las figuras, donde cada píxel es una variable notable. Por ejemplo, una capa totalmente

unida para una figura (pequeña) de dimensión 100 x 100 tiene 10,000 pesos para cada neurona en la segunda capa. La operación de convolución otorga una satisfacción a esta duda, ya que minimiza la cantidad de parámetros disponibles, admitiendo que la red sea más honda con menos parámetros. Por ejemplo, independientemente de la dimensión de la figura, las regiones de mosaico de dimensión 5 x 5, cada una con iguales pesos distribuidos, solicitan solo 25 parámetros que se pueden estudiar. Mediante el deterioro de pesos regularizados sobre menos parámetros, se obvian las dudas de gradiente de escape y detonación de gradiente analizados durante la dispersión hacia atrás en redes neuronales habituales.

Validación de un modelo

Al momento de aprobar una técnica de enseñanza o prototipo es obligatorio afirmar que se trabajará eficientemente para las referencias de ensayo, de manera que pueda coger la duda del problema a solucionar y difundir adecuadamente. En el fondo evita ser ajeno a las referencias cogidas durante su preparación, eludiendo la duda conocido como sobre entrenamiento.

La sobre entrenamiento se puede fijar, inconsistentemente, como el riesgo que corremos al sobreentrenar un prototipo.

Consiste en que este concluya contestando rigurosamente a los dominios del entrenamiento de referencia de preparación y que sea insuficiente de emplearse con equilibrio de destreza apropiado a otros entrenamientos de referencias que logren surgir en un futuro.

En la figura 16, deseamos señalar que cuando platicamos de referencias de preparación y de test, estamos haciendo alusión en concesión a los algoritmos de ejercitamiento inspeccionado, ya que es obligatorio valorar las soluciones logradas sobre referencias marcadas jamás verificado anteriormente y se pueden equiparar los fallos encargados para cada grupo.

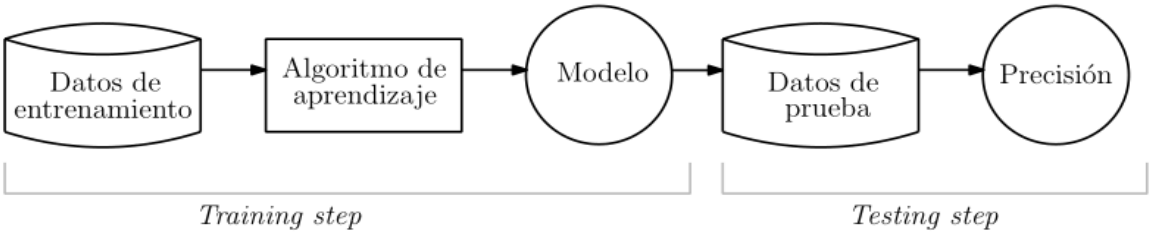


Figura 16.Técnica orientada en aprendizaje supervisado.

Fuente: (Procesando imágenes, 2014)

Así, utilizaremos el grupo de referencias de preparación para inventar el prototipo controlado, ya que el grupo de referencias de test se utilizará para calcular la exactitud esperada por el prototipo. Configurará una porción del desarrollo de edificación del prototipo, la reproducción repetitiva de preparación y comprobación inclusive lograr fases de exactitud y de amplitud de exactitud favorable.

Matríz de confisión

Esta sección muestra en un cuadro una percepción esquemática de los fallos atribuidos por el prototipo de distribución. Se inicia en un prototipo esquematizado para enfocar la etapa de acierto de un prototipo de exactitud. Ya que se le conoce como cuadro de emergencia.

La figura 17 muestra a detalle la distribución binaria de los elementos.

		Clase predicha	
		P	N
Clase verdadera	P	TP	FN
	N	FP	TN

Figura 17.Resumen de confusión binaria.

Fuente: (Procesando imágenes,2014)

Validación cruzada k-fold

Radica en coger las referencias originales e inventar a partir de ellos dos grupos liberados: el primer grupo de preparación (y prueba), y un segundo grupo de validación.

Después, el grupo de preparación se va a fraccionar en k subgrupos y, al instante de hacer la preparación, se va a coger cada k subgrupo como grupo de prueba del prototipo, entretanto que la diferencia de las referencias se cogerá como grupo de preparación.

Este desarrollo se reiterará k veces, y en cada repetición se cogerá un grupo de prueba diferente, mientras las referencias sobrantes se utilizarán, como se citó, como grupo de preparación. Una vez concluida las repeticiones, se evalúa la exactitud y el fallo para cada uno de los prototipos creados, y para lograr la exactitud y el fallo final se evalúa el promedio de los k prototipos preparados.

Una vez se calcula con esta exactitud promedio para un prototipo, se puede reiterar entonces la técnica del Cross Validation para todos los demás prototipos de distribución que se estén calculando, y se cogerá al final aquel que realice el mejor valor de exactitud y menor fallo promedio.

Por lo tanto, podemos coger dicho prototipo sobre el grupo de validación producido en la parte inicial, ya que, se supone, es este prototipo el que mejor solución en absoluto ofreció durante el ciclo de preparación.

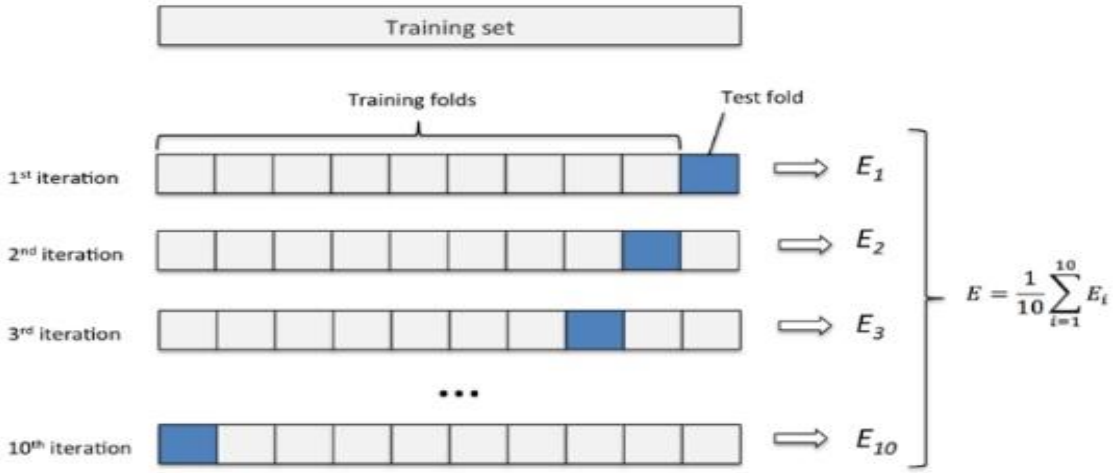


Figura 18. Resumen de validación cruzada K-Fold.

Fuente: (Procesando imágenes,2014)

Curva ROC

Calcula la utilidad de la relación a los incorrectos positivos y auténticos positivos. La transversal de la circunferencia de la curva se da a través de un prototipo creado casualmente, ya que los valores bajos se estiman malos que una evaluación casual de referencias.

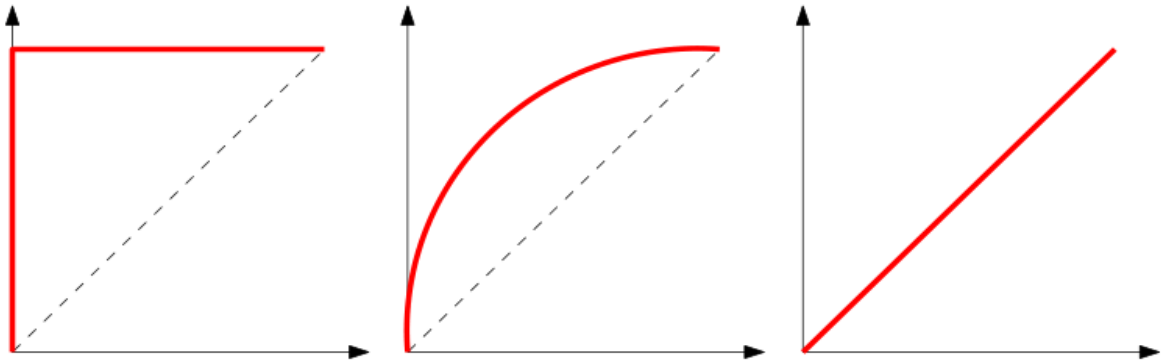


Figura 19. Resumen de curvas ROC. Fuente: (Procesando imágenes,2014)

1.3.7. Lenguaje de programación Python

Permite y utiliza un estilo de determinación universal la cual es muy potente y dócil, a la vez que simple y fácil de aprender. Es un lenguaje de elevado nivel,

que tolera marchar sencillamente todo tipo de configuración de referencias, tanto numéricos como de texto.

Simplificado y rápido

Este lenguaje facilita demasiado la programación “origina que te acomodes a una manera de lenguaje de programación, Python te sugiere un patrón”. Es un gran lenguaje para scripting, si usted pide algo ágil (en el sentido de la ejecución del lenguaje), con unas pocas líneas de código ya está aclarado.

Elegante y flexible

Python brinda varios instrumentos, si usted desea listas de diversos tipos de referencias, no hace falta que explicar cada tipo de referencias. Es un lenguaje tan dócil usted no se inquieta tanto por las definiciones.

Programación sana y productiva

Python se transforma en un procedimiento muy robusto de sistematizar: es fácil de instruirse, dirigido a los ajustes correctos, le hace como dependiente de mejorar, ejecutar las reglas, la utilización de las líneas, de variables. Por consiguiente, es un lenguaje que fue formado con rendimiento en mente, es decir, Python le permite ser más productivo, le permite otorgar en los tiempos que me soliciten.

Ordenado y limpio

La disposición que sostiene Python, es de lo que más les agrada a las personas, es muy claro, cualquier otro programador lo puede interpretar y laborar sobre el proyecto escrito en Python. Los patrones están bien estructurados, a diferencia de otros lenguajes.

Portable

Python muy sencillo (sea en Mac, Linux o Windows) en semejanza con otros lenguajes. No obstante, las librerías no tienen la necesidad de ser instalarlas adicionalmente como en otros lenguajes.

Comunidad

Una nota muy significativa para el avance de un lenguaje es la sociedad, la misma sociedad de Python cuida el lenguaje y casi el total de los reajustes se realizan de manera democrática.

Librerías

Numpy: Es la principal librería y nos ofrece grandes sistemas de datos, matrices multidimensionales.

OS: Nos apoya a listar contenidos de una carpeta.

RE: Usa la sintaxis Perl

Matplotlib: Otorga esquemas a partir de la contención de datos

Keras: Nos facilita a inventar arquetipos rápidamente con redes neuronales

1.4. Formulación del Problema

¿De qué manera se podrá realizar la identificación de grietas en pistas de asfalto utilizando procesamiento de imágenes?

1.5. Justificación e importancia del estudio

En nuestros días es común localizar en nuestros barrios cualquier rajadura en las pistas ya que se debe al salitre, las lluvias, las malas prácticas en construcción de pistas de asfalto y la circulación de tránsito pesado lo cual permite el avance de daños en toda la pista. Sin embargo, el mal estado de nuestras pistas asfaltadas es incalculable lo cual atribuye a que el tránsito terrestre sea lento en la mayoría de nuestras ciudades. Provocando un malestar enorme y un retraso en nuestras actividades diarias cuando decidimos trasladarnos a nuestro centro de labores o casas de estudios. Originando la incertidumbre, la depreciación de casas, el retraso económico dentro de nuestra sociedad.

Así mismo, las rajaduras en las pistas de asfalto son las principales causas que hacen que nuestras pistas se deterioren mucho más rápido con el pasar del tiempo. Para solucionar este problema se propone utilizar algoritmos que ayuden a identificar las grietas de manera automática en beneficio de nuestra sociedad.

Esto ha sido el punto de partida que me ha motivado a la ejecución de esta investigación haciendo el uso del procesamiento digital de imágenes a través de algoritmos.

1.6. Hipótesis

La identificación automática de grietas en pistas asfaltadas se podrá realizar automáticamente sí utilizamos un método de procesamiento de imágenes digitales y algoritmos de aprendizaje.

1.7. Objetivos

1.7.1. Objetivo general

Realizar la identificación automática de grietas en pistas de asfalto utilizando procesamiento digital de imágenes.

1.7.2. Objetivos específicos

- **a.** Establecer un protocolo de adquisición de imágenes de grietas de pistas de asfalto.
- **b.** Seleccionar los algoritmos de clasificación.
- **c.** Procesar las imágenes de grietas de asfalto
- **d.** Implementar el algoritmo seleccionado
- **e.** Realizar un análisis de los resultados obtenidos

II. MÉTODO

2.1. Tipo y Diseño de Investigación

2.1.1. Tipo de Investigación

Para la presente investigación será Cuantitativa Tecnológica Aplicada porque el software a desarrollar será beneficioso para el campo del reconocimiento de imágenes digitales en ambientes no controlados.

2.1.2. Diseño de Investigación

El tipo de la investigación es cuasi experimental ya que la propuesta de este presente trabajo de investigación realizará pruebas con las imágenes sin contar con un grupo de control establecido. Se tomarán todas las imágenes y se les procesa **para comparar y analizar resultados.**

2.2. Población y muestra

2.2.1. Población

La población definida por 500 imágenes grietas.

2.2.2. Muestra

La muestra ha sido determinada por conveniencia y se ha determinado que será el 80% de la población, es decir; un total de 400 imágenes. Pero es debido a que otras investigaciones (Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks). Se han obtenido resultados satisfactorios con esta cantidad de imágenes.

2.3. Variables, Operacionalización

2.3.1. Variable independiente

Procesamiento digital de imágenes.

2.3.2. Variable dependiente

Identificación de grietas

2.3.3. Operacionalización de variables

Tabla 1.

Operacionalización de variables de la Investigación

Variables	Indicadores	Fórmula
	Error	$ERR=\frac{F_P+F_N}{F_P+F_N+T_P+T_N}$
Variable Dependiente Identificación de grietas	Exactitud (E)	(VP+VN)/(VP+VN+FP+FN)
	Sensibilidad (S)	VP/(VP+FN)
	Especificidad	VN/(FP+VN)
	AUC	(1/2) * (S+E)
Variable Independiente Procesamiento digital de imágenes	Tiempo de respuesta	$T_R=T_A-T_I$

Nota: Elaboración propia

2.4. Técnicas e instrumentos de recolección de datos, validez y confiabilidad.

Tabla 2.

Recolección de datos - Técnicas.

Descripción	Forma de aplicación	Forma de obtención
Observación	Personal	Para verificar el rendimiento de los experimentos realizados con las aplicaciones a desarrollar.
	Instrumentos	
Ficha de observación	Personal	Los datos se obtienen después de la evaluación para medir los diferentes indicadores se registran para luego ser analizados.
Variable Independiente Procesamiento digital de imágenes	Tiempo de respuesta	

Nota: Elaboración propia.

2.5. Procedimiento de análisis de datos

Observación: Se obtuvieron las imágenes con la ayuda de una cámara para captar fotografías. Las imágenes tomadas fueron seleccionadas para poder determinar cuales estaban óptimas, es decir que no estén borrosas, ni tomadas a contraluz, ni muy oscuras; para luego realizar el procesamiento de reconocimiento de imágenes.

2.6. Criterios éticos

Social: El presente proyecto ayudaría a reducir el mal estado de pistas de asfalto, generando una excelente comunicación ininterrumpida y brindando un mejor servicio a la población.

Responsabilidad: Este proyecto de investigación se le considera de manera responsable porque prevendría el mal estado de pistas de asfalto y evitaría las colisiones entre vehículos (accidentes de tránsito terrestre) y la “no comunicación” entre ciudades, debido a que se realizaría un mantenimiento preventivo a las pistas de asfalto y de este modo, se aseguraría un alto índice de comunicación ininterrumpida.

2.7. Criterios de Rigor Científico

Originalidad

Cumplir con citar las fuentes bibliográficas de las cuales se ha utilizado para llevar a cabo la investigación.

Validez

Los datos obtenidos de la investigación y sus resultados son evaluados y analizados por especialistas en el tema para confirmar su validez.

Consistencia

La investigación muestra material consistente y certificado por la comunidad científica.

III. RESULTADOS

3.1. Resultados en Tablas y Figuras

El resultado de la ejecución de la propuesta obtuvo los siguientes datos la cual se evidencia en la Tabla 3.

Tabla 3.

Matriz de confusión de resultados.

Variables	**Con grietas (0)**	**Sin grietas (1)**	
Clasificó Con grietas (0)	*VN =22*	*FP =1*	23
Clasificó Sin grietas (1)	*FN =1*	*VP =8*	9
	23	*9*	

Nota: Elaboración propia.

a) Exactitud = (VP+VN) / (VP +VN+ FP + FN)

= (8+22) / (8 +22+ 1 + 1)

= 0.94 = 94%

b) Sensibilidad = VP/(VP + FN)

= (8)/(8+1)

= 0.88 = 88%

c) Especificidad = VN/(FP + VN)

= 22/(1 + 22)

= 0.96 = 96%

d) AUC = (1/2)*(S+E)

= (1/2)*(0.88+0.94)

= 0.91 = 91%

e) Error = 1-Exactitud

= 1-0.94

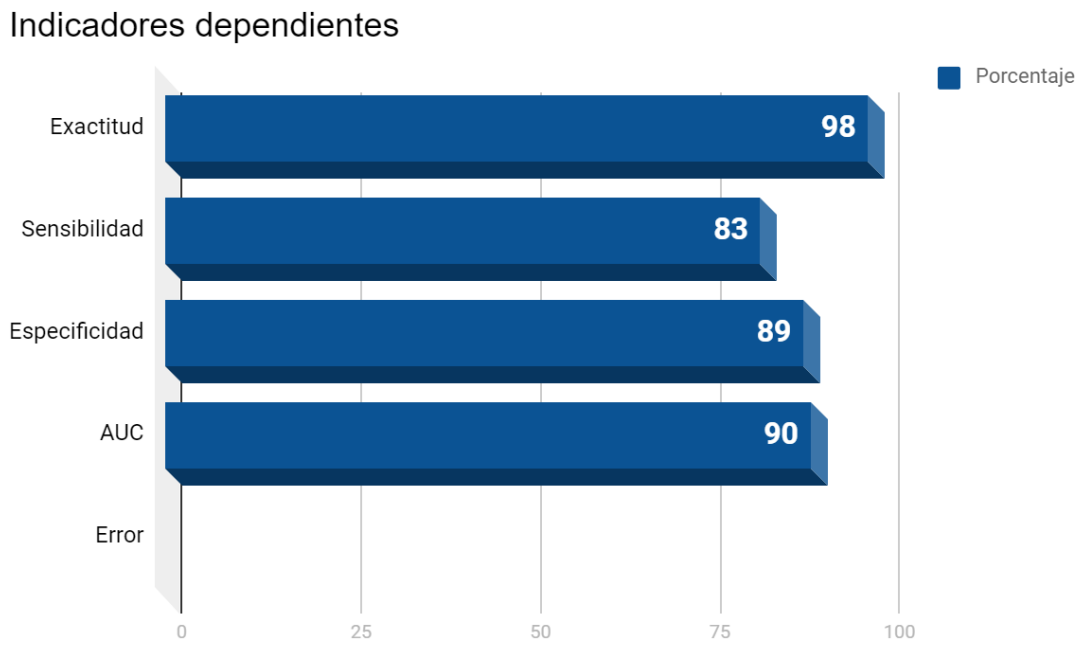
=0.1 = 0.1%

f) Tiempo de respuesta = TA-TI = 68.28 segundos considerando el tiempo de entrenamiento y validación

Variables dependientes:

Tabla 3.

Resultados para los indicadores de la variable dependiente (exactitud, sensibilidad, especificidad, AUC y error)



Nota: (Elaboración propia)

En el indicador de exactitud obtuvo 98% significa que el modelo de red convolucional propuesto sirve para clasificar grietas de asfalto en imágenes, debido a que el porcentaje es mayor a 93%.

En el indicador de sensibilidad obtuvo un 83%, significa que dio positivo un 83% en grietas de asfalto del 100% de pruebas con grietas, 17% de imágenes con grietas de asfalto no fueron correctamente clasificadas.

En el indicador de especificidad obtuvo un 89%, significa que dio negativo un 89% de grietas de asfalto del 100% de pruebas sin grietas, 11% de imágenes sin grietas de asfalto no fueron correctamente clasificadas.

Tabla 4.

Rango de clasificación de la Curva ROC para su interpretación.

Rango de clasificación	
Rango AUC	Clasificación
90<=AUC<=100	Excelente
80<=AUC<90	Bueno
70<=AUC<80	Regular
60<=AUC<70	Malo

Nota: Tomado de (Zhu et al., 2010)

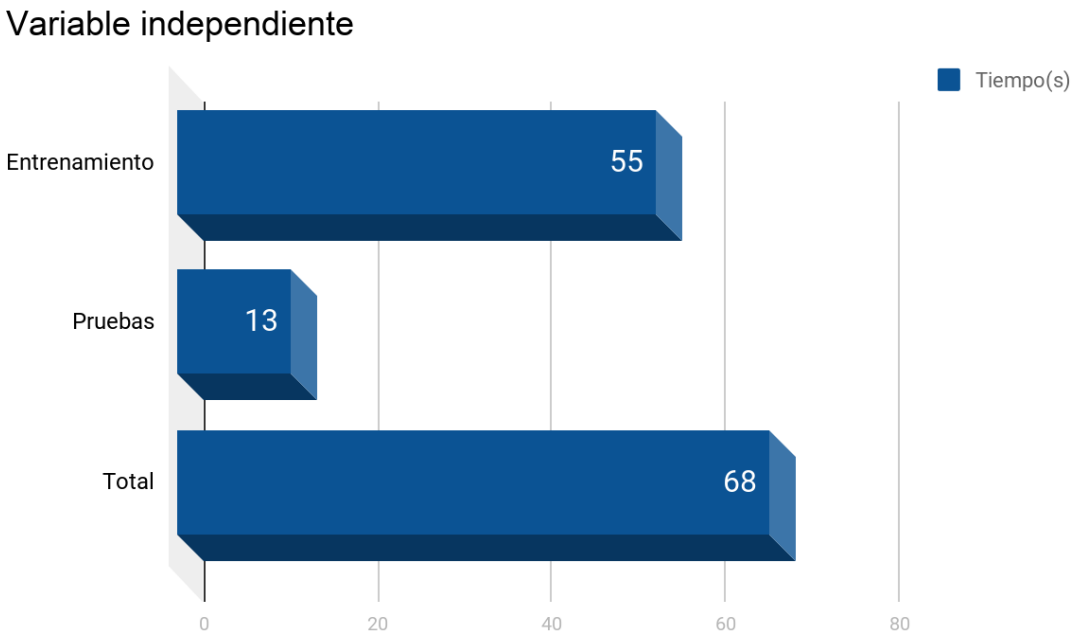
En el indicador AUC obtuvo un 90% teniendo en cuenta el rango de clasificación de la Tabla 4 da una clasificación “Excelente”.

En el indicador de Error obtuvo 0.03%, significa que menos del 1% de las imágenes del dataset de pruebas no fueron clasificadas correctamente.

Variables independientes:

Tabla 5.

Resultados para el indicador tiempo para la variable independiente.



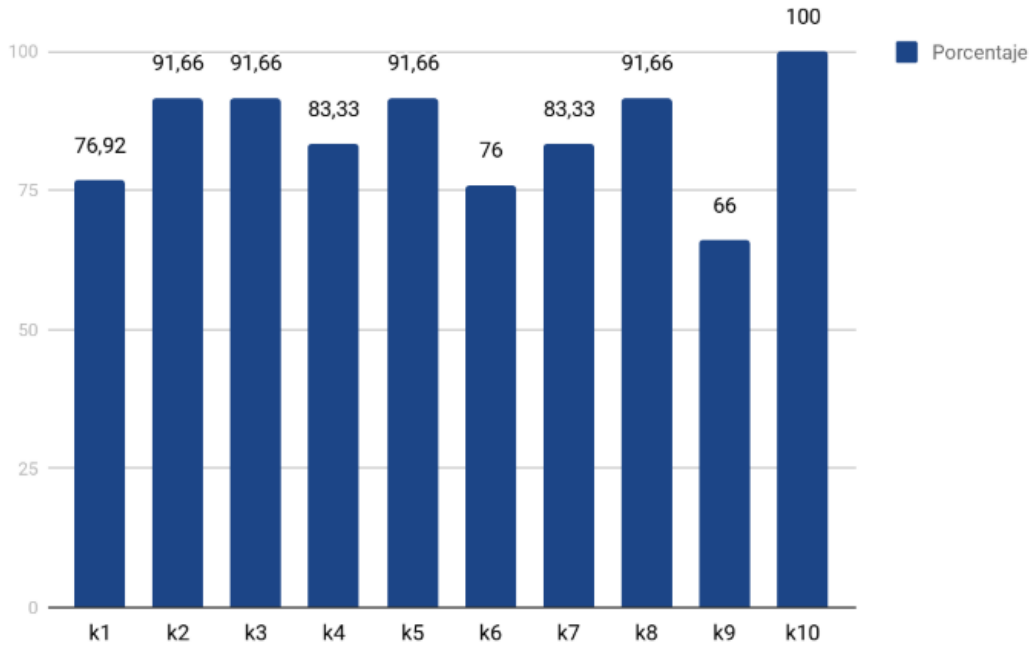
Nota: (Elaboración propia)

3.2. Discusión de resultados

En los trabajos de (Cha et al., 2017), (Tong et al., 2017), (Chen & Jahanshahi, 2018), (Silva & Lucena, 2018) utilizaron un marco de deep learning basado en redes neuronales convolucionales (CNN) para la detección de grietas por lo cual el algoritmo escogido es CNN por su eficacia en la detección de grietas.

Tabla 6.

Exactitud utilizando validación cruzada k-fold.



Nota: (Elaboración propia)

Exactitud utilizando validación cruzada k-fold.

La exactitud del modelo fue de 98% al analizar los resultados de indicadores obtenido, el algoritmo CNN obtuvo unos resultados similares a lo analizado por el conjunto de datos de prueba 17% de imágenes de grietas de asfalto fueron clasificadas como sin grietas y 11% de imágenes sin grietas fueron clasificadas con grietas.

A comparación de las investigaciones citadas en los antecedentes de estudio. Observamos que una investigación realizada por (Yong HU, 2014) utilizaron un fresco enfoque de localización de rajaduras en el asfalto de manera instantánea originada en el estudio de trama y detalle de formas. Equiparando con localizador de bordes habituales, las soluciones efectivas enseñaron que todas las

rajaduras son reveladas satisfactoriamente por la técnica sugerida, aun con base de trama difícil o con alumbrado disparejo. Los resultados obtenidos alcanzan un porcentaje del 70% donde se demuestran que las imágenes muestran una grieta transversal en la superficie del pavimento con iluminación desigual. En algunas regiones tienen un alto brillo, fuerte fondo de textura. La imagen es discontinua en su estructura. Además, muestra una ligera grieta longitudinal en la superficie. Los resultados demuestran que la textura compleja, iluminación desigual y no uniforme. Con estos antecedentes es un gran desafío desarrollar este sistema de detección y evaluación de dificultades en el asfalto.

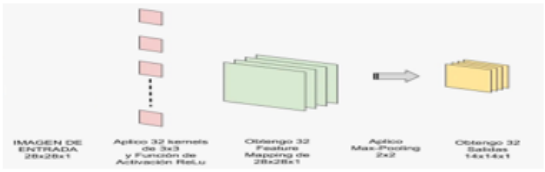
Otra de las investigaciones es la desarrollada por (Ma, Wang, Zha, Di y Zhu, 2014) en su artículo “Pavement Cracks Detection Based on FDWT”; Manifiesta una nueva opción en la localización de defectos en asfaltos por medio de la fracción diferencial y transformada de wavelets. El empleo de la fracción diferencial permite arreglar los indicadores de media y de gran frecuencia Posteriormente se emplea la FDWT para filtrar la distorsión que enseña unida a la umbralización acondicionada para la división y localización de lugares en donde existan rajaduras. El método propuesto alcanza el 70%, donde las imágenes de pavimento con altos niveles de textura superficial que dificulta la detección de grietas.

3.3. Aporte práctico.

La presente investigación, pretende proporcionar un método de algoritmo para la ejecución de tratamientos cuyo efecto provoca una relevancia considerable en el desarrollo de la segmentación y así mismo generar algoritmos con más efectividad donde es posible incrementar la precisión y reducir la dificultad computacional.

3.3.1.- ESTABLECER UN PROTOCOLO ADQUISICIÓN DE IMÁGENES DE GRIE

3.3.4.- IMPLEMENTAR EL ALGORITMO SELECCIONADO



3.3.2.- SELECCIONAR LOS ALGORITMOS DE CLASIFICACIÓN

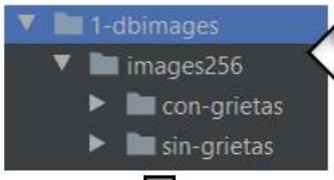
No.	Algorithm/class	1	2	3	4	5	6	7	8	9	Total
1	Global thresholding	4	2	1	1	1	1	1	1	1	13
2	Fractal analysis	4	3	2	1	1	1	1	2	1	16
3	Quatersion edge detection	2	1	3	3	1	1	1	4	1	17
4	FMC	1	1	3	3	3	1	1	1	4	18
5	SVM	3	4	3	2	2	1	1	1	1	18
6	Dynamic thresholding	3	4	3	2	2	1	1	1	2	19
7	Region splitting	3	3	4	2	2	1	1	2	1	19
8	Canny edge detection	3	4	3	1	2	1	2	2	1	19
9	Wavelet transform	3	1	3	4	3	1	1	1	2	19
10	Region merging/growing	3	3	4	2	2	1	1	2	2	20
11	Region merging and splitting	3	3	2	2	3	1	1	1	4	20
12	Clustering	2	3	3	4	1	1	1	3	2	20
13	Graph (MST)	3	3	3	3	1	1	1	1	4	20
14	Valley edge detection	2	3	1	2	1	1	4	4	3	21
15	Fractional differential	2	3	2	3	1	1	4	3	2	21
16	Multiple scales	3	3	2	3	1	1	3	4	1	21
17	Neural network	2	2	2	2	2	2	3	3	4	22
18	Deep learning	2	2	2	4	2	2	4	3	3	24

CNN

(Red Neuronal Convolucioal)

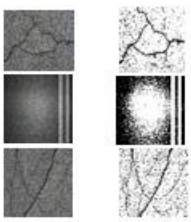
3.3.5.- REALIZAR UN ANÁLISIS DE LOS RESULTADOS OBTENIDOS

3.3.3.- PROCESAR LAS IMÁGENES DE GRIETAS DE ASFALTO



Dataset de grietas sin procesar

Aplicar filtro thresholding binario



Dataset de grietas etiquetadas

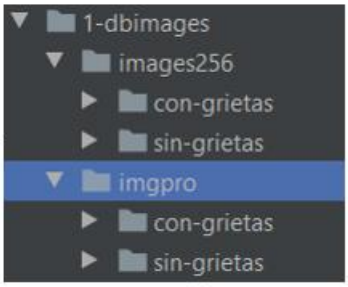


Figura 20. *Método Propuesto. Fuente:* (Elaboración propia)

3.3.1. Establecimiento del protocolo de adquisición de imágenes de grietas de pistas de asfalto.

En este escenario, para la extracción de las imágenes de pistas de asfaltos, se diseñó el stick y un brazo metálico de soporte para la cámara y además de activar el temporalizador de la misma para tomar imágenes cada 5 segundos.

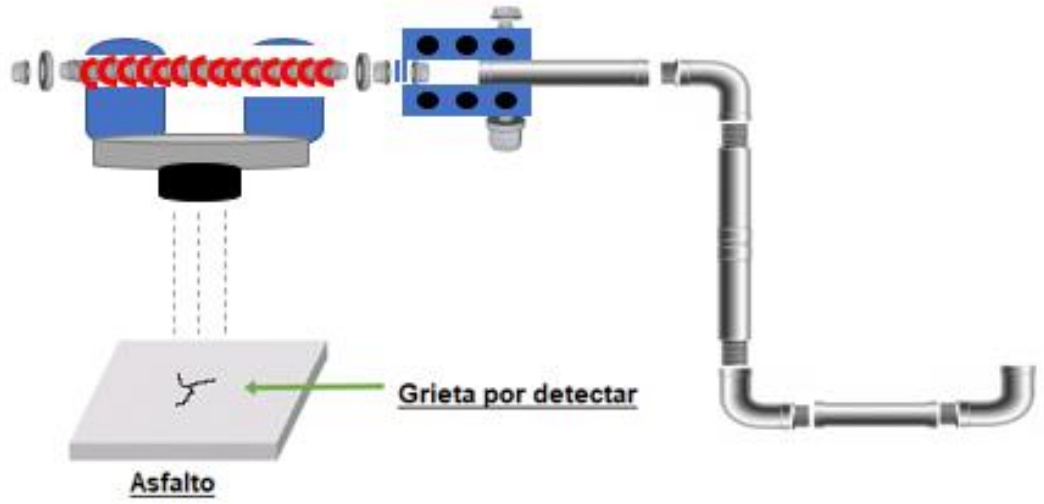


Figura 21. Construcción del stick y del brazo mecánico. Fuente *(elaboración propia)*

Para construir el brazo metálico se utilizaron los siguientes componentes:

Fabricar 5 piezas de metal:

En este apartado fabriqué 5 piezas de metal, donde cada pieza está enumerada. Posteriormente, para su ensamble cada pieza cuenta con un hilo del tipo rosca que sirve para unir una pieza tras otra y poder armarla y desarmarla sin dificultad.

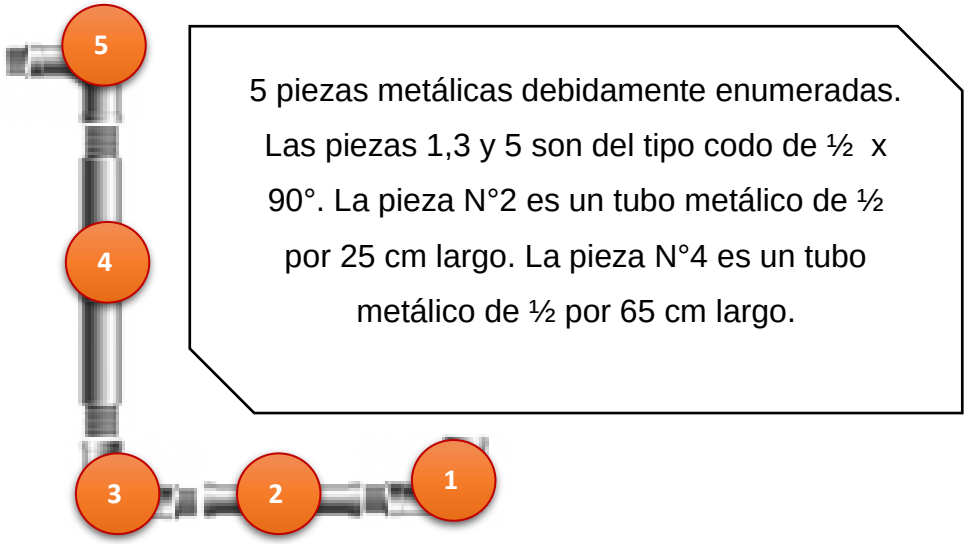


Figura 22. *Brazo metálico enumerado. Fuente*: (Elaboración propia)

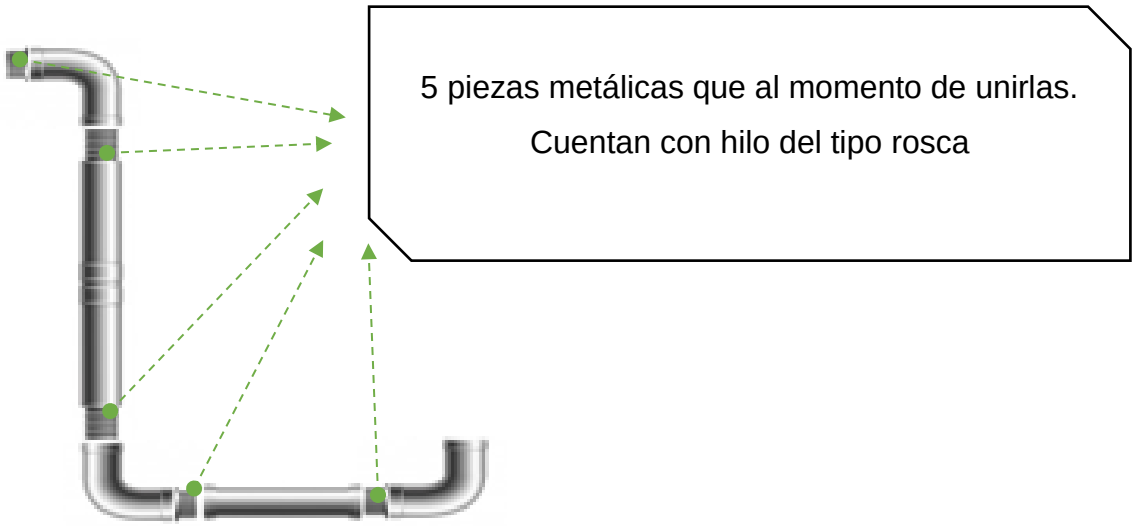


Figura 23. *Brazo metálico armado*. *Fuente:* (Elaboración propia)

Fabricar el Stick:

Los materiales para este apartado son: 2 tubos metláticos de ½ por 5cm de largo, 4 anillos de ¼ , 2 anillos de presión de ¼ , una tuerca de ¼ , un perno de ¼ x 2 pulgadas de largo, una lámina de metal perforada de 13 cm de largo x 5 cm de ancho, un resorte de 5 cm de largo y por último un tubo metálico de ½ x 15 cm largo.

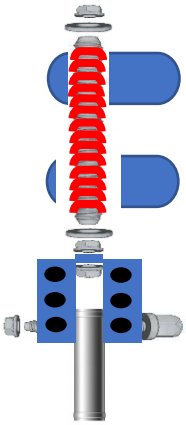


*Figura 24.*Fabricación del Stick.

Fuente: *(Elaboración propia)*

Cámara para adquirir imágenes:

La cámara digital es de la marca Sony Cybershot, zoom óptico 4x,gran angular 26 mm, 14.4 mega pixeles.

Figura 25. Cámara propia para adquirir imágenes.

Fuente: *(Elaboración propia)*

Características de mi ordenador con la cual estoy realizando mi investigación.

Es una laptop HP Envy 15, Intel Core i3-3630QM 2.4GHz (3.2GHz c/TB), Pantalla LED 15.6" Full HD (1920x1080), 8GB RAM / Disco Duro 1TB.

Figura 26. Cámara propia para adquirir imágenes. Fuente: *(Elaboración propia)*

Creación de la Dataset

Un Dataset es una agrupación de referencias y se tienen el total de las estimaciones que posee cada componente ya sea la altitud y el peso del elemento que afecta a cada sujeto de la agrupación de referencias. Estos componentes se les denomina referencia. La agrupación de referencias alberga datos para uno o más individuos en acto de su cantidad de filas como se evidencia en la Figura 29.

3.3.2. Selección de los algoritmos de clasificación

En este apartado de la presente investigación se ha hecho una matriz para seleccionar los algoritmos de clasificación

Según (Wang et al., 2019) se analizó 18 algoritmos de detección de grietas para 9 tipos de grietas en imágenes, en la escala de malo (1), regular (2), bueno (3), excelente (4) y las ordenó según su eficiencia en la detección.

Tabla 7. *Algoritmos de detección de grietas según su eficacia.*

No.	Algorithm/class	1	2	3	4	5	6	7	8	9	Total
1	Global thresholding	4	2	1	1	1	1	1	1	1	13
2	Fractal analysis	4	3	2	1	1	1	1	2	1	16
3	Quaternion edge detection	2	1	3	3	1	1	1	4	1	17
4	FMC	1	1	3	3	3	1	1	1	4	18
5	SVM	3	4	3	2	2	1	1	1	1	18
6	Dynamic thresholding	3	4	3	2	2	1	1	1	2	19
7	Region splitting	3	3	4	2	2	1	1	2	1	19
8	Canny edge detection	3	4	3	1	2	1	2	2	1	19
9	Wavelet transform	3	1	3	4	3	1	1	1	2	19
10	Region merging/growing	3	3	4	2	2	1	1	2	2	20
11	Region merging and splitting	3	3	2	2	3	1	1	1	4	20
12	Clustering	2	3	3	4	1	1	1	3	2	20
13	Graph (MST)	3	3	3	3	1	1	1	1	4	20
14	Valley edge detection	2	3	1	2	1	1	4	4	3	21
15	Fractional differential	2	3	2	3	1	1	4	3	2	21
16	Multiple scales	3	3	2	3	1	1	3	4	1	21
17	Neural network	2	2	2	2	2	2	3	3	4	22
18	Deep learning	2	2	2	4	2	2	4	3	3	24

Nota: (Wang et al,2019).

Las redes neuronales y deep learning forman parte del machine learning, actualmente, proporcionan la mejor solución para la clasificación y la verificación de imágenes. En las redes neuronales, las neuronas artificiales forman las unidades de cálculo necesarias, y las interconexiones entre ellas se utilizan para describir la red. En el aprendizaje profundo, se utilizan múltiples capas de unidades de procesamiento no lineales para la conversión y extracción de características. Puede

representar conceptos en múltiples formas jerárquicas correspondientes a diferentes niveles de abstracción.

Para la clasificación y el reconocimiento de grietas en el pavimento, se pueden aplicar redes neuronales y aprendizaje profundo. Para la identificación de imágenes, es posible que se deba configurar manualmente la estructura de los datos de imágenes de grietas, y los datos deben estar en un formato particular en orden. Por ejemplo, se configuran dos carpetas, una para entrenamiento y otra para pruebas. En el conjunto de entrenamiento, la clasificación de imágenes es un proceso muy tedioso. En algunos casos, la tarea de entrenamiento se vuelve casi imposible cuando se necesita etiquetar una gran cantidad de imágenes, pero los datos de entrenamiento son muy importantes para el aprendizaje profundo. Si hay una gran cantidad de imágenes en el conjunto de entrenamiento, el modelo de identificación de imágenes con grietas tiene una mejor oportunidad de reconocimiento de imágenes. (Wang et al., 2019).

De acuerdo a la tabla y al análisis que se ha realizado se determinó trabajar con Deep Learning.

3.3.3. Adquisición de las imágenes de grietas de asfalto

Para poder realizar mi trabajo de tesis en plena pandemia, tuve que buscar la manera de como plasmar mis conociemientos adquiridos y llevarlos a la realidad, ya que para ese momento nos encontrábamos en cuarentena por lo que estaba prohibido salir y más aún con vehículo. Para este trabajo se realizó la adquisición de imágenes con ayuda del brazo metálico 500 imágenes, cada imagen presentó una resolución de 5888×3584 pixeles. de acuerdo a (Cha et al., 2017) para identificar grietas de asfaltos usando CNN se debe dividir la imagen de una gran resolución en imágenes de poca resolución para un mejor estudio.

Luego se le aplicó el thresholding binario.

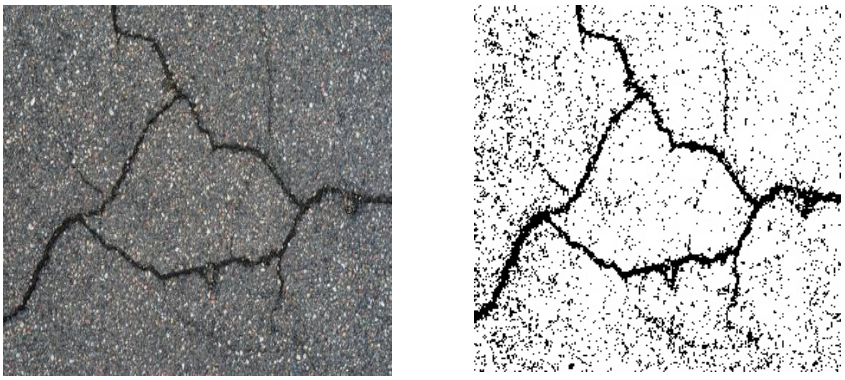


Figura 27. *Procesar la imagen y aplicando thresholding binario. Fuente:* (Elaboración propia)

Se dividió cada imagen original en imágenes pequeñas con resolución 256x256 píxeles, generando un total de 4 por cada imagen original, dando un total de 160 imágenes, estas imágenes conforman el Dataset a utilizar por el algoritmo de detección.

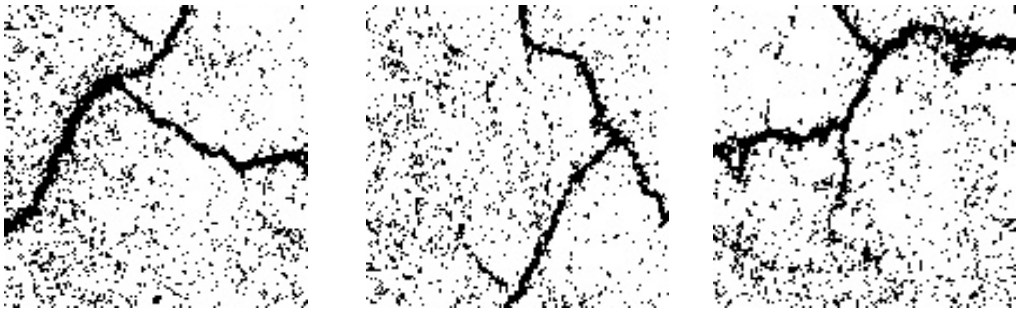


Figura 28. *Recortes de imágenes de 256x256 pixeles. Fuente:* (Elaboración propia).

Luego procedió a etiquetar las imágenes, se separó las imágenes que presentaban grietas de las que no presentaban ninguna grieta en carpetas

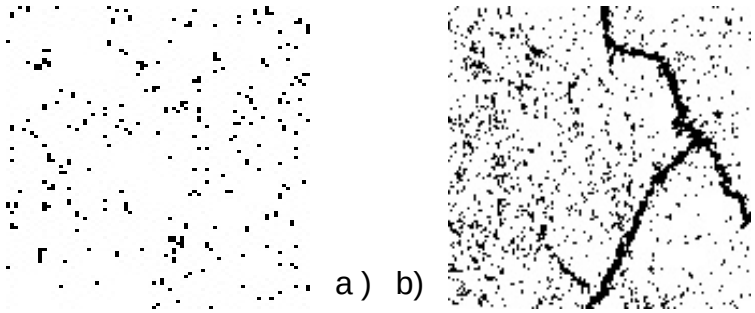


Figura 29. *Imágenes sin grietas (a) y con grietas(b). Fuente:* (Elaboración propia)

A partir de las imágenes de 256x256 píxeles se procede la creación del dataset:

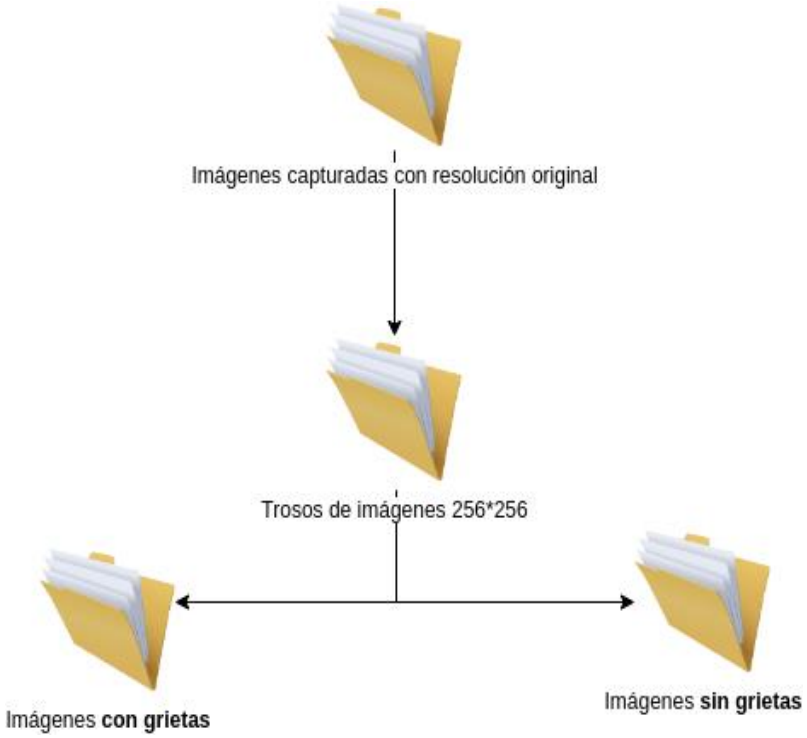
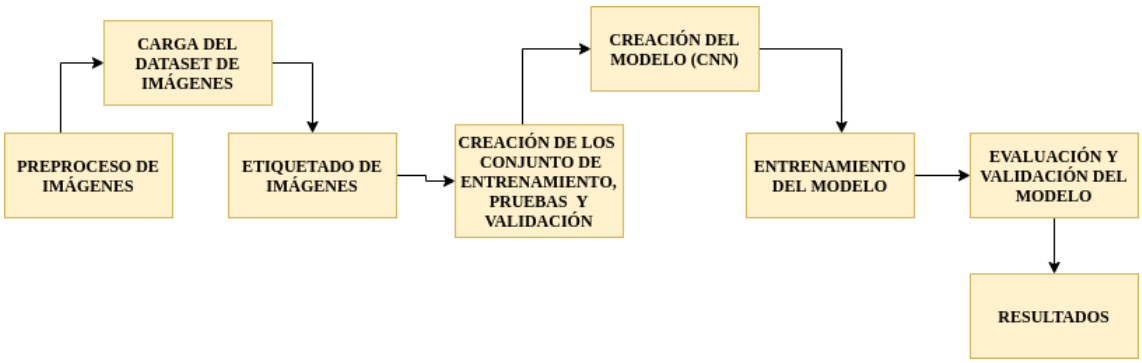


Figura 30. DataSet. Fuente: *(Elaboración propia)*

3.3.4. Implementación del algoritmo seleccionado

Tabla 8. *Diagrama de bloques del método propuesto*



Nota: (Elaboración propia)

1. Pre proceso de imágenes

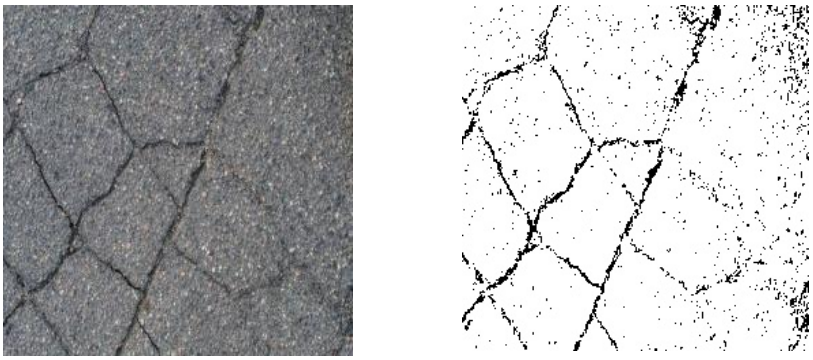


Figura 31. *Imágenes en memoria Fuente:* (Elaboración propia)

Utilizando la librería opencv, se carga el dataset de imágenes a memoria, se aplica el filtro thresholding binario a todas las imágenes para reducir la textura del asfalto, y se exportan las imágenes.

```
def conversion_images(path_imgs, path_save):
    :param path_imgs: Dirección fisica de la carpeta de imagenes a tratar
    :param path_save: Dirección fisica de la carpeta donde se va a guardar
    list_names = get_list_names(path_imgs)  # Extrae la lista de nombres
    for name in list_names:  # recorre la lista de nombres
        path_image = f'{path_imgs}{name}'  # path fisico de cada imagen
        img_read = cv2.imread(path_image, 0)
        alpha = 1.1  # Control de contraste simple
        beta = 30  # Control de brillo simple
        img_read =  alpha* img_read + beta
        cv2.imwrite(f'{path_save}{name}', img_read)
        # global thresholding binary
        ret1, th1 = cv2.threshold(img_read, 120, 255, cv2.THRESH_BINARY)
        cv2.imwrite(f'{path_save}{name}' , th1)
        print(f'Imagen procesada {name} : {th1.shape}')
```

Fuente: Elaboración propia

2. Carga del dataset de imágenes

Teniendo en cuenta el dataset de imágenes preprocesadas en el punto anterior las cuales conforman el dataset para realizar la propuesta planteada, para ello se carga en memoria las imágenes redimensionadas con y sin grietas utilizando la librería nativa para el lenguaje python “os” y “matplotlib” para establecer la ruta y posterior carga de cada imagen de un directorio.

```
imagenes = []

directorios = []

cant_directorios = []

ruta_anterior = ''

cant = 0

tiempo_inicio = time()

# Funición para calcular el tiempo de ejecución la cual recibe dos parametros

# hora de fin y la hora de inicio calcula la diferencia y lo transforma en segundos y lo devuelve

print("leyendo imagenes de ", ruta_imagen)

for raiz, nombres_dir, nombres_archivos in os.walk(ruta_imagen):

    for nombre_archivo in nombres_archivos:

        if re.search("\.(jpg|jpeg|png|bmp|tiff)$", nombre_archivo):

            cant = cant + 1

            ruta_archivo = os.path.join(raiz, nombre_archivo)

            imagen = plt.imread(ruta_archivo)

            imagenes.append(imagen)

            b = "Leyendo..." + str(cant)

            print(b, end="\r")
```

```
        if ruta_anterior != raiz:
            print(raiz, cant)
            ruta_anterior = raiz
            directorios.append(raiz)
            cant_directorios.append(cant)
            cant = 0
cant_directorios.append(cant)
cant_directorios = cant_directorios[1:]
cant_directorios[0] = cant_directorios[0] + 1
print('Directorios leidos:', len(directorios))
print("Imagenes en cada directorio", cant_directorios)
print('Suma total de imagenes en subdirs:', sum(cant_directorios))
```

Fuente: Elaboración propia

3. Etiquetado de imágenes

Se establece las etiquetas con valores de 0 o 1 las cuales corresponden a la clase con grietas y sin grietas respectivamente. Esto se realiza con el fin de emplear un método controlado y señalar que cuando se carga una imagen correspondiente a la clase con grietas está automáticamente queda etiquetada con el valor «0» y con esta información de entrada y salida esperada el modelo puede entrenar y ajustar el conocimiento y mejor la clasificación.

```
# Creamos las etiquetas
etiquetas = []
indice = 0
for cantidad in cant_directorios:
    for i in range(cantidad):
        etiquetas.append(indice)
    indice = indice + 1
print("Cantidad etiquetas creadas: ", len(etiquetas))
grietas = []
indice = 0
for directorio in directorios:
    nombre = directorio.split(os.sep)
    print(indice, nombre[len(nombre) - 1])
    grietas.append(nombre[len(nombre) - 1])
    indice = indice + 1
y = np.array(etiquetas)
X = np.array(imagenes, dtype=np.uint8)  # convierto de lista a numpy
# Encuentra los números únicos de las etiquetas
clases = np.unique(y)
cant_clases = len(clases)
print('Número total de salidas : ', cant_clases)
print('Clases de salida : ', clases)
```

Fuente: Elaboración propia

4. Creación del conjunto de entrenamiento, pruebas y validación

Se utiliza los píxeles de las imágenes como el arreglo de entrada, y las etiquetas como la salida esperada de la red neuronal convolucional. Posteriormente fraccionó el dataset en el 80% para entrenamiento y 20% para pruebas, luego se utilizó 20% del dataset de entrenamiento para validación como se evidencia en la Figura 40.

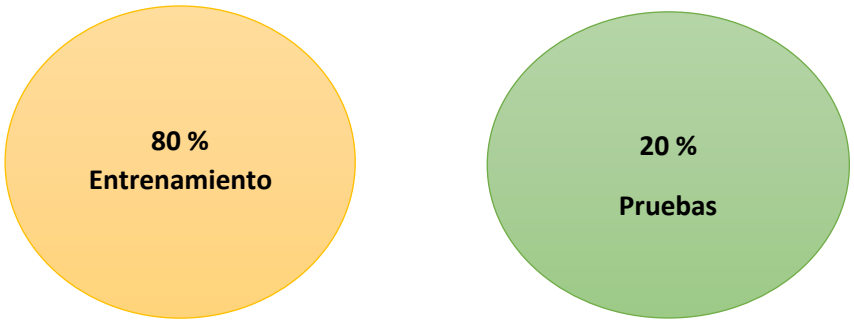


Figura 32. Grupo de entrenamiento, validación y prueba de la información. Fuente: *(Elaboración propia)*

```
# Creamos Sets de Entrenamiento y Test

train_X, test_X, train_Y, test_Y = train_test_split(X, y, test_size=0.2)

print('Datos para entrenamiento : ', train_X.shape, train_Y.shape)

print('Datos para pruebas : ', test_X.shape, test_Y.shape)

# A contnuación ya pasamos a preprocesar las imagenes para el diseño de la red convolucional

train_X = train_X.reshape(train_X.shape[0], 256, 256, 1)

test_X = test_X.reshape(test_X.shape[0], 256, 256, 1)

train_X = train_X.astype('float32')

test_X = test_X.astype('float32')

train_X = train_X / 255.

test_X = test_X / 255.

# Hacemos el One-hot Encoding para la red
```

```
# Change the etiquetas from categorical to one-hot encoding
train_Y_one_hot = to_categorical(train_Y)
test_Y_one_hot = to_categorical(test_Y)
# Visualice el cambio para la etiqueta de categoría utilizando codificación de uno en caliente
print('Etiqueta original:', train_Y[0])
print('Después de la conversión a one-hot:', train_Y_one_hot[0])
# Creamos el Set de Entrenamiento y Validación
# Mezclar todo y crear los grupos de entrenamiento y testing
train_X, valid_X, train_label, valid_label = train_test_split(train_X,
                                                              train_Y_one_hot,
                                                              test_size=0.2,
                                                              random_state=13)
print(train_X.shape, valid_X.shape, train_label.shape, valid_label.shape)
```

Fuente: Elaboración propia

5. Creación del modelo (CNN)

El modelo utilizado para realizar la propuesta es una Red Neuronal Convolucional (CNN) la cual manifiesta una fila de ventajas muy optimista:

a)Orden jerárquico para estudiar escenarios de abstracción.

b)Facultad par estudiar ejercicios complicados

c)Estudiar representaciones de propiedades directas en forma espóntanea a partir de la información mínima la cual se modeló utilizando la librería de inteligencia artificial Keras 2.3.1, la cual presenta la siguiente arquitectura:

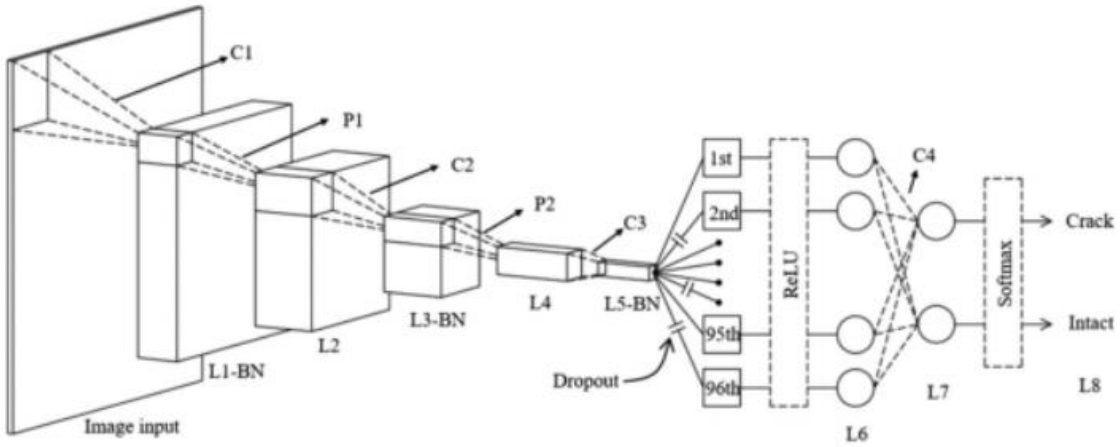


Figura 33. Arquitectura de la red neuronal convolucional. Fuente: *(Young-Jin Cha, Wooram Choi)*

Este modelo (CNN) se encuentran integradas de diversos filtros/núcleos, los cuales establecen un grupo de datos entrenables y que logran convolucionar singularmente una figura obtenida para hallar particularidades dadas como orilla y siluetas.Esta elevada cantidad de fitros se logra emplear para capturar particularidades especiales de la figura en servicio de los pesos memorizados. No obstante, se logra reducir con triunfo una figura establecida en una representación elevadamente abstracta que es sencilla de anunciar.

Aplicar fórmula

$$\frac{dx^c}{dt} = -x^c(t) + a_d^c\, y^d(t) + b_d^c\, u^d + i$$

$$y^c = \frac{1}{2}\ (|x^c\,(t) + 1| - |x^c\,(t) - 1|\,)$$

Modelo matemático de la CNN

Donde:

xc es la celda, uc es la entrada y yc es la salida.

```
# Creamos el modelo de CNN

# declaramos variables con los parámetros de configuración de la red

INIT_LR = 1e-3  # Valor inicial de learning rate. El valor 1e-3 corresponde con 0.001

epochs = 100  # Cantidad de iteraciones completas al conjunto de imagenes de entrenamiento

batch_size = 100  # cantidad de imágenes que se toman a la vez en memoria

grietas_model = Sequential()

# priemra capa

grietas_model.add(Conv2D(24, kernel_size=(3, 3), input_shape=(256, 256, 1)))

grietas_model.add(MaxPooling2D((7, 7)))

# segunda capa

grietas_model.add(Conv2D(48, kernel_size=(3, 3)))

grietas_model.add(MaxPooling2D((4, 4)))

# tercera capa

grietas_model.add(Conv2D(96, kernel_size=(3, 3)))

grietas_model.add(Activation('relu'))

# cuarta capa

grietas_model.add(Conv2D(2, kernel_size=(3, 3), input_shape=(1, 1)))

grietas_model.add(Flatten())

grietas_model.add(Dense(2, activation='softmax'))

grietas_model.summary()

grietas_model.compile(loss=keras.losses.categorical_crossentropy,

                      optimizer=keras.optimizers.Adagrad(lr=INIT_LR,

                                                         decay=INIT_LR / 100), metrics=['accuracy'])
```

Model: "sequential_1"

Layer (type)	Output Shape	Param#
conv2d_1 (Conv2D)	(None,254,254,24)	240
max_pooling2d_1 (MaxPooling2)	(None,36,36,24)	0
conv2d_2 (Conv2D)	(None,34,34,48)	10416
max_pooling2d_2 (MaxPooling2)	(None,8,8,48)	0
conv2d_3 (Conv2D)	(None,6,6,96)	41568
activation_1 (Activation)	(None,6,6,96)	0
conv2d_4 (Conv2D)	(None,4,4,2)	1730
flatten_1 (Flatten)	(None,32)	0
dense_1 (Dense)	(None,2)	66

Total params: 54,020

Trainable params: 54,020

Non-trainable params: 0

Fuente: Elaboración propia

6. Entrenamiento del modelo

Se entrena la red neuronal convolucional con el 64% del Dataset de imágenes, con un valor inicial constante de 100 épocas correspondiente a las iteraciones completas por conjunto de imágenes de entrenamiento y con una cantidad máxima de 50 imágenes a la vez en memoria. Este proceso se llevó a cabo utilizando una máquina (Sistema: Windows10, Procesador: Intel(R) Core (TM) i3-6560U CPU @ 2.2. GHz, Memoria instalada (RAM): 8.0 GB, Tipo de sistema: Sistema operativo de 64 bits, procesador x64). El cual tardó un tiempo promedio de 65 segundos

Entrenamos el modelo: Aprende a clasificar imágenes

este paso puede tomar varios minutos, dependiendo de tu ordenador, cpu y memoria ram libre

como ejemplo, en mi pc tarda 4 minutos

grietas_train = grietas_model.fit(train_X, train_label, batch_size=batch_size, epochs=epochs, verbose=1,

validation_data=(valid_X, valid_label))

Fuente: Elaboración propia

7. Evaluación y validación del modelo

7.1 Evaluación

Se evalúa la red neuronal convolucional propuesta mediante los indicadores de exactitud, sensibilidad, especificidad, auc y error usando agrupación de referencia de prueba, dividiendo la cantidad de imágenes correctamente clasificadas, ya sea con grietas o sin grietas, entre el total de imágenes.

```
Y_pred = grietas_model.predict_classes(test_X)
df = pd.DataFrame(Y_pred)
print(df)
print('Matriz de confusión')
matrix = confusion_matrix(test_Y, Y_pred)
df = pd.DataFrame(matrix)
df.index = ['con_getas', 'sin_grietas']
df.columns = ['con_gririetas', 'sin_grietas']
print(df)
#Calculo de indicadores
tn, fp, fn, tp = matrix.ravel()
print((tn, fp, fn, tp))
exactitud = (tp + tn) / (tn + fp + fn + tp)
sensibilidad = tp / (tp + fn)
especificidad = tn / (tn + fp)
auc = 0.5 * (exactitud + sensibilidad)
error = 1 - exactitud
tiempo_transcurrido = time() - tiempo_inicio
print(
    f' Exactitud: {exactitud:.2f}\n '
    f'Sensibilidad:{sensibilidad:.2f}\n '
    f'Especificidad: {especificidad:.2f}\n '
    f'AUC: {auc:.2f}\n '
    f'Error: {error:.2f}\n '
```

f'Tiempo: {tiempo_transcurrido:.2f}')

Fuente: Elaboración propia

7.2. Validación

La validación del se realizó empleando el procedimiento de la validación cruzada k-fold, ya que se encuentra verificado teóricamente donde la eficacia no cambia significativamente si agregamos un valor más. Esta técnica genera una valoración porcentual por cada ejecución de acuerdo al indicador de exactitud es decir que genera 10 resultados porcentuales de los cuales se tomó el valor promedio.

```
# Model configuration

nro_epochs = 100  # Cantidad de iteraciones completas al conjunto de imagenes de entrenamiento

batch_size = 50  # cantidad de imágenes que se toman a la vez en memoria

validation_split = 0.2

verbosity = 1

num_folds = 10

# Definir los contenedores de puntuación por fold

acc_per_fold = []

loss_per_fold = []

# Fusionar las entradas y los objetivos

inputs = np.concatenate((train_X, valid_X), axis=0)

targets = np.concatenate((train_label, valid_label), axis=0)

# Definir el validador de K-fold Cross Validator

kfold = KFold(n_splits=num_folds, shuffle=True)

# Evaluación del modelo de validación cruzada de K-fold
```

```
fold_no = 1
for train, test in kfold.split(inputs, targets):
  # Generar una impresión
  print(f'Entrenamiento para fold {fold_no} ...')
  grietas_model = creacion_modelo()
  # Ajustar los datos al modelo
  history = grietas_model.fit(inputs[train], targets[train],
                              batch_size=batch_size,
                              epochs=nro_epochs,
                              verbose=verbosity,
                              validation_split=validation_split)
  # Generar métricas de generalización
  scores = grietas_model.evaluate(inputs[test], targets[test], verbose=0)
  print(
    f'Puntuación por fold {fold_no}: {grietas_model.metrics_names[0]} de {scores[0]}; '
    f'{grietas_model.metrics_names[1]} hasta {scores[1] * 100}%')
  acc_per_fold.append(scores[1] * 100)
  loss_per_fold.append(scores[0])
  grietas_model.save(f'model_binary_{fold_no}.h5py')
  # Aumentar el número de folds
  fold_no = fold_no + 1
# == Proporcionar las puntuaciones medias ==
for i in range(0, len(acc_per_fold)):
```

```
    print(f'> Fold {i + 1} - Pérdida: {loss_per_fold[i]} - Exactitud: {acc_per_fold[i]}%')

print('Puntuaciones para todos los folds:')

print(f'> Exactitud: {np.mean(acc_per_fold, dtype=np.float32)} (+- {np.std(acc_per_fold, dtype=np.float32)})')

print(f'> Pérdida: {np.mean(loss_per_fold)}')
```

Fuente: Elaboración propia

3.3.5. Análisis de los resultados obtenidos

En este apartado, podemos verificar que las “pruebas con grietas” (P: con grietas). Es satisfactorio y semejante en los resultados con grietas (R: con grietas).

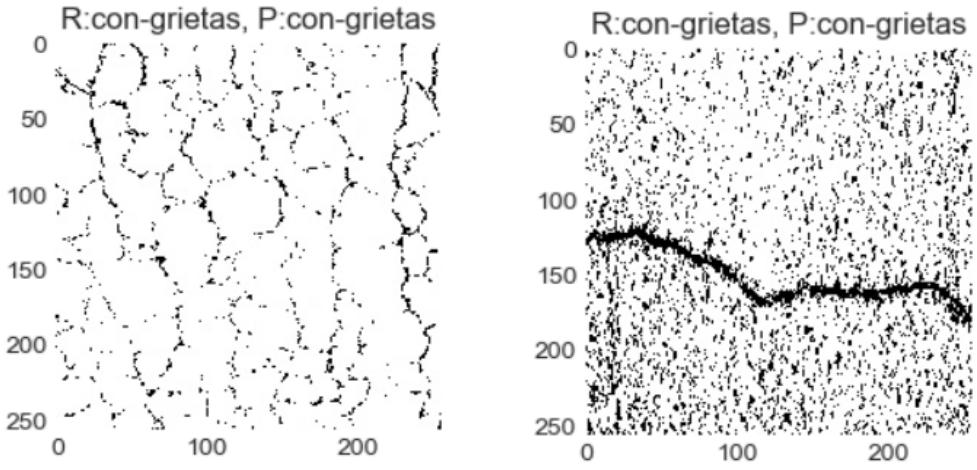


Figura 34.*Pruebas y resultado. Fuente:* (Elaboración propia)

Aquí observamos que las iteraciones al set de entrenamiento, se logra un valor del 98%. Esto quiere decir que el algoritmo va ganando conocimiento mientras pasen las épocas de información.

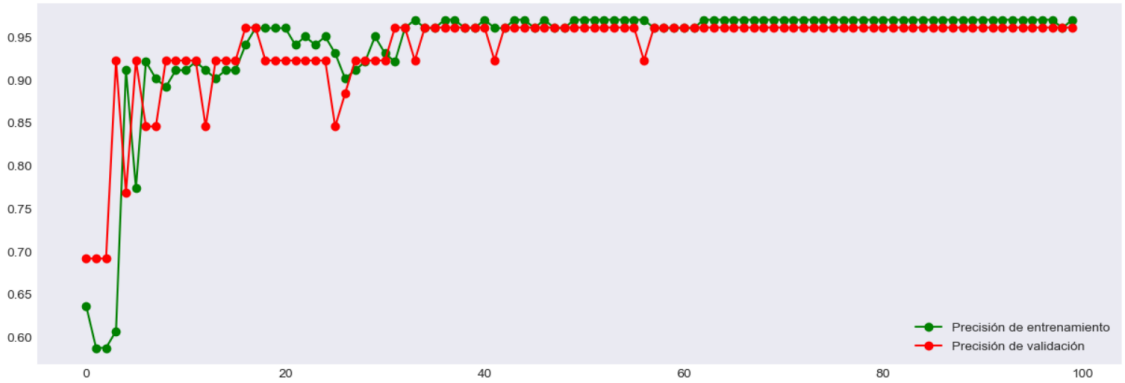


Figura 35. *Iteraciones de entrenamiento*. *Fuente:* (Elaboración propia)

Al inicio es regular la pérdida. A medida que aumentan las épocas de entrenamiento va ganando información. Donde la pérdida de entrenamiento va a disminuir.

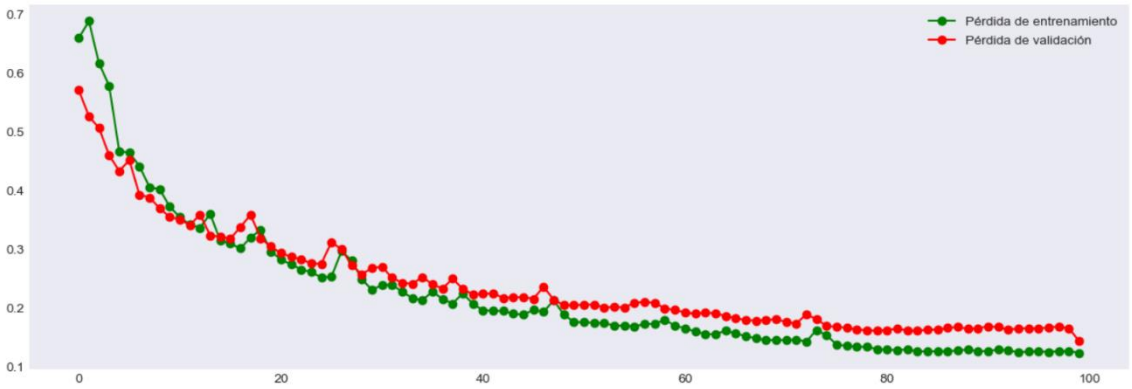


Figura 36. Iteraciones de pérdida. Fuente: *(Elaboración propia)*

IV. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

1. La construcción del brazo metálico, me permitió captar las imágenes de grietas en pistas de asfalto.

2. El algoritmo de Red Neuronal Convolucional (CNN) tuvo buena performance en los trabajos de investigación relacionados en la clasificación de grietas, por tal motivo se escogió este algoritmo.

3. El procesamiento de las imágenes, aumentando el brillo y aplicando el filtro umbralización binario, permitió visualizar mejor las grietas de asfalto y esto mejoró la identificación del modelo.

4. El desarrollo de la propuesta se ejecutado empleando el lenguaje de programación Python el cual ha reunido y ha ganado aceptación en la innovación de aplicaciones vinculadas a la ciencia de datos, en cuanto a las soluciones logradas se debe finalizar que satisface todos los objetivos propuestos empleando los algoritmos seleccionados.

5. La realización de la propuesta demuestra que de los resultados alcanzados para el algoritmo de CNN obtuvo 98% de exactitud en tiempo promedio de 85 segundos.

4.2. Recomendaciones

1. El rendimiento del algoritmo de clasificación CNN depende del trabajo de investigación que aplica el algoritmo, por lo que si no se prueba en el mismo entorno de investigación, se recomienda no asumir que da buenos resultados.

2. Se recomienda utilizar otros filtros, tales como Otsu y Canny Edges para experimentar otros resultados.

3. Si bien es cierto Python, es un lenguaje enfocado en ciencia de datos, pero no es el único que se utiliza para este propósito, se recomienda utilizar otras tecnologías basadas en Java como Weka o en C# como Machine Learning de Microsoft Azure.

4. Se recomienda tener una mayor población de imágenes de asfalto con grietas y sin grietas para lograr una mayor precisión en la clasificación.

V. REFERENCIAS.

ADEX PERÚ. (20 de Agosto de 2018). *Lambayeque se consolida como región líder en producción de capsicum*. Obtenido de Asociación de Exportadores-ADEX: http://www.adexperu.org.pe/notadeprensa/lambayeque-se-consolida-como-region-lider-en-produccion-de-capsicum/

Agraria. (29 de Agosto de 2018). *La exportación de ajíes peruanos llegaría a valores de US$ 6 millones al cierre de este año*. Obtenido de agraria.pe: https://agraria.pe/noticias/la-exportacion-de-ajies-peruanos-llegaria-a-valores-de-us-6--17338

Agraria.pe. (05 de Junio de 2018). *Exportaciones de peruanas de capsicum alcanzarían los US$ 246 millones este año*. Obtenido de Asociacion de Gremios Productores Agrarios del Perú- AGAP: https://agapperu.org/noticias/exportaciones-de-peruanas-de-capsicum-alcanzarian-los-us-246-millones-este-ano/

Alwaseela, A., Haiyan, C., Ahmed, E.-m., & Yong, H. (Julio de 2019). Infield oilseed rape images segmentation via improved unsupervised learning models combined with supreme color features. Computers and Electronics in Agriculture. *Elsevier*, 1057-1068.

APEGA,UNALM,INIA &USMP. (2009). *Ajíes peruanos sazón para el mundo.* Lima: Sociedad Peruana de Gastronomía (APEGA). Obtenido de https://www.adexperu.org.pe/capsicum/documents/ajiesdelPeru.pdf

B. Xu and Y. Huang. (2015). 3-11.

Basu, S., & Krishna, A. (2003). *Capsicum, The genus Capsicum* (I ed.). (A. K. De, Ed.) Estados Unidos y Canada: CRC Press.

Bellocchio E., Constante G., Cascianelli S., Fravolini M., & Valigi P. (2020). Combining Domain Adaptation and Spatial Consistency for Unseen Fruits Counting: A Quasi-Unsupervised Approach. *IEEE Xplore*, 1-8.

Bosland, P. (1999). *Encyclopedia of Chiles.* Nueva York: B.Hanson.

Bosland, P. (2001). *Encyclopedia of chiles* (VII ed.). Nueva York: B.hanson.

Carreteras viales. (2014).

Charu, A. (2014). *Data Classification Algorithms and Applications* (Vol. I). Estados Unidos: Chapman and Hall/CRC.

Chen, S., Shivakumar, S., Dcunha, S., Das, J., Okon, E., Qu, C., . . . Kumar, V. (2017). Counting Apples and Oranges With Deep Learning: A Data-Driven Approach. *IEEE Xplore*, 1-8.

Chi, C., Dinh, T., Hoang, D., Quoc, B., & Quoc, D. (2017). Automatic dragon fruit counting using adaptive thresholds for image segmentation and shape analysis. *IEEE Xplore*, 132-137.

Clifford, P., & Cosma, I. (2011). A Statistical Analysis of Probabilistic Counting Algorithms. Scandinavian Journal of Statistics. *Scandinavian Journal of Statistics*, 1-14.

Construcción de asfalto. (2014).

Contingencia vial. (2014). Lima.

Córdova, N. (28 de Junio de 2019). *Día del Cebiche: ajíes regionales imprescindibles para este embajador culinario peruano*. Obtenido de Agencia Peruana de Noticias: https://andina.pe/agencia/noticia-dia-del-cebiche-ajies-regionales-imprescindibles-para-este-embajador-culinario-peruano-756980.aspx

Devi, G., Neelamegam, P., & Sudha, S. (2017). Image Processing System for Automatic Segmentation and Yield Prediction of Fruits using Open CV. *IEEE Xplore*, 758-762.

Diario el correo. (2019). *Pistas en mal estado.* Lima.

Dirección General de Investigación y Estudios Sobre Comercio Exterior. (2018). *Reporte de Comercio Regional Lambayeque.* Lambayeque: MINCETUR. Obtenido de https://www.mincetur.gob.pe/wp-content/uploads/documentos/comercio_exterior/estadisticas_y_publicaciones/estadisticas/reporte_regional/Mensual/RMCR_noviembre_2018.pdf

El comercio. (08 de Septiembre de 2018). *Adex plantea restituir la aplicación de detracciones al sector capsicum*. Obtenido de El comercio: https://elcomercio.pe/economia/peru/adex-plantea-restituir-aplicacion-detracciones-sector-capsicum-noticia-nndc-555358-noticia/

Estructura pavimental. (2014).

Gestión. (04 de Junio de 2008). *Exportaciones de Capsicum crecerían más de 3% y cerrarían el año en US$ 246 millones*. Obtenido de GESTIÓN:

https://gestion.pe/economia/exportaciones-capsicum-crecerian-3-cerrarian-ano-us-246-millones-235188-noticia/

gob.pe. (08 de Setiembre de 2019). *En el Día Nacional de los Ajíes, pequeños productores son los principales exponentes*. Obtenido de Plataforma Digital Única del Estado Peruano: https://www.gob.pe/institucion/minagri/noticias/50645-en-el-dia-nacional-de-los-ajies-pequenos-productores-son-los-principales-exponentes

INEI. (15 de Junio de 2020). *Producción Nacional.* Obtenido de Instituto Nacional de Estadística e Informática –INEI: http://m.inei.gob.pe/media/MenuRecursivo/boletines/informe-tecnico-produccion_abril-2020.pdf

INGE CUC. (2012). 7-10.

J. Zhu. (2015).

La república. (2019). Pistas asfaltadas y deterioradas en el perú. *La república*, 5-7.

Lal, S., Behera, S., Sethy, P., & Kumar, A. (2017). Identification and Counting of Mature Apple Fruit Based on BP Feed Forward Neural Network. *IEEE Xplore*, 361-368.

Liu, X., Chen, S. A., Sivakumar, N., Dcunha, S., Qu, C., Taylor, C., . . . Kumar, V. (2018). Robust Fruit Counting: Combining Deep Learning, Tracking, and Structure from Motion. *IEEE Xplore*, 1045-1052.

Long-Solís, J. (1998). *Capsicum y cultura: la historia del chilli.* México: FCE - Fondo de Cultura Económica. Obtenido de https://elibro.net/es/ereader/bibsipan/110320?page=87

Makkar, T., Yogesh, & Jothi, A. (2018). A comparative analysis of different time bounded segmentation techniques. *IEEE Xplore*, 32-37. doi:10.1109/GUCON.2018.8675104.

Manual de carreteras Mantenimiento Vial. (2014).

Morton, C. (2009). *Image Analysis, Classification, and change detection in remote sensing.* Estados Unidos: CRC Press.

MTC. (julio de 2016). *Pistas del Perú.* Lima.

Ponce, J., Aquino, A., Borja, M., & Andujar, J. (2019). Automatic Counting and Individual Size and Mass Estimation of Olive-Fruits Through Computer Vision Techniques. *IEEE Access*, 59451-59465.

Revista CUC. (2016). *Revista CUC*, 15-17.

RPP. (04 de Septiembre de 2017). *Adex: Cada peruano come 4.75 kilos de ají al año, pero mexicanos el doble*. Obtenido de RPP Noticias: https://rpp.pe/economia/economia/adex-cada-peruano-come-475-kilos-de-aji-al-ano-pero-mexicanos-el-doble-noticia-1074496?ref=rpp

Salazar, I., Pacheco, A., Rodriguez, R., Lezama, J., & Huamán, S. (2019). An image processing method to automatically identify Avocado leaf state. *IEEE Xplore*, 1-5.

Schueller, J., He, Y., & Zhang, Q. (2021). Infield Oilseed Rape Images Segmentation via Improved Unsupervised Learning Models Combined with Supreme Color Features. *Computers and Electronics in Agriculture*, 1057-1068.

SUTRAN. (2016). *Inspección de pavimentos.* LIMA.

Townes, M. S. (2014). California.

Yazgaç, B., & Mürvet, K. (2019). Fractional order calculus based fruit detection. *IEEE Xplore*, 1-4.

Yogesh, & Dubey, K. (2016). Fruit Defect Detection Based on Speeded Up Robust Feature Technique. *IEEE Xplore*, 590-594.

Z. Wan-zhi, 2013. (2013).

ANEXOS

Anexo 1: Resolución de aprobación de tesis

USS | UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
RESOLUCIÓN N° 0273-2022/FIAU-USS

Pimentel, 12 de mayo de 2022

VISTOS:
El Acta de reunión N°1105-2022 del Comité de investigación de la Escuela profesional de INGENIERÍA CIVIL remitida mediante oficio N°0125 -2022/FIAU-IS-USS de fecha 11 de mayo de 2022, y;

CONSIDERANDO:

Que, de conformidad con la Ley Universitaria N°30220 en su artículo 48º que a letra dice: "La investigación constituye una función esencial y obligatoria de la universidad, que la fomenta y realiza, respondiendo a través de la producción de conocimiento y desarrollo de tecnologías a las necesidades de la sociedad, con especial énfasis en la realidad nacional. Los docentes, estudiantes y graduados participan en la actividad investigadora en su propia institución o en redes de investigación nacional o internacional, creadas por las instituciones universitarias públicas o privadas.";

Que, de conformidad con el Reglamento de grados y títulos en su artículo 21° señala: "Los temas de trabajo de investigación, trabajo académico y *tesis* son *aprobados por el Comité de Investigación* y derivados a la facultad o Escuela de Posgrado, según corresponda, para la emisión de la resolución respectiva. El *periodo de vigencia de los mismos será de dos años*, a partir de su aprobación. En caso un tema perdiera vigencia, el Comité de Investigación evaluará la ampliación de la misma.

Que, de conformidad con el Reglamento de grados y títulos en su artículo 24º señala: La tesis es un estudio que debe denotar rigurosidad metodológica, originalidad, relevancia social, utilidad teórica y/o práctica en el ámbito de la escuela profesional. Para el grado de doctor se requiere una tesis de máxima rigurosidad académica y de carácter original. Es individual para la obtención de un grado; *es individual o en pares para obtener un título profesional*. Asimismo, en su artículo 25° señala: "El tema debe responder a alguna de las líneas de investigación institucionales de la USS S.A.C.".

Que, según documentos de vistos el Comité de investigación de la Escuela profesional de INGENIERÍA DE SISTEMAS acuerda aprobar el(los) tema(s) de Tesis, así como aprobar la designación de asesor y/o jurados y ampliar vigencia de tema de tesis a cargo de los estudiantes o egresados que se detallan en el anexo de la presente Resolución.

Estando a lo expuesto, y en uso de las atribuciones conferidas y de conformidad con las normas y reglamentos vigentes;

SE RESUELVE:

ARTÍCULO 1°: APROBAR, el tema de las Tesis perteneciente a la línea de investigación de INFRAESTRUCTURA, TECNOLOGÍA Y MEDIO AMBIENTE, a cargo de los egresados del Programa de estudios de **INGENIERÍA DE SISTEMAS** según se detalla en el anexo de la presente Resolución.

ARTÍCULO 2°: APROBAR, la designación de Asesor especialista y/o Jurado evaluador en el extremo del tema de la tesis quedando tal como se detalla en el anexo de la presente Resolución.

ARTÍCULO 3°: AMPLIAR VIGENCIA, de tema de las Tesis perteneciente a la línea de investigación de INFRAESTRUCTURA, TECNOLOGÍA Y MEDIO AMBIENTE, a cargo de los egresados del Programa de estudios de **INGENIERÍA DE SISTEMAS** según se detalla en el anexo de la presente Resolución.

ARTÍCULO 4°: DEJAR SIN EFECTO, toda Resolución emitida por la Facultad que se oponga a la presente Resolución.

ANEXO

UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
RESOLUCIÓN N° 0273-2022/FIAU-USS

Pimentel, 12 de mayo de 2022

IGNACIO SOTO PERCY ROBUSTIANO	IDENTIFICACIÓN AUTOMÁTICA DE LAS GRIETAS EN PISTAS DE ASFALTO UTILIZANDO PROCESAMIENTO DIGITAL DE IMÁGENES	1874-2019/FIAU-USS	HASTA EL 31 DE DICIEMBRE DEL 2022

Mg. Victor Alexci Tuesta Monteza
Decano (e) / Facultad De Ingeniería, Arquitectura Y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

Mg. Carlos William Atalaya Urrutia
Secretario Académico / Facultad de ingeniería, arquitectura y urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

REGÍSTRESE, COMUNÍQUESE Y ARCHÍVESE

Cc: Interesado, Archivo

Anexo 2: Protocolo de adquisición de imágenes

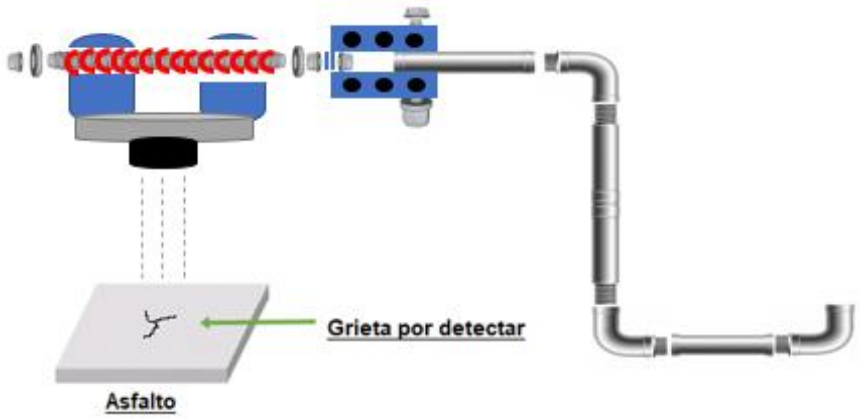


Figura 21. Construcción del stick y del brazo mecánico.

Fuente: *Elaboración propia*

Anexo 3: Construcción del Brazo mecánico

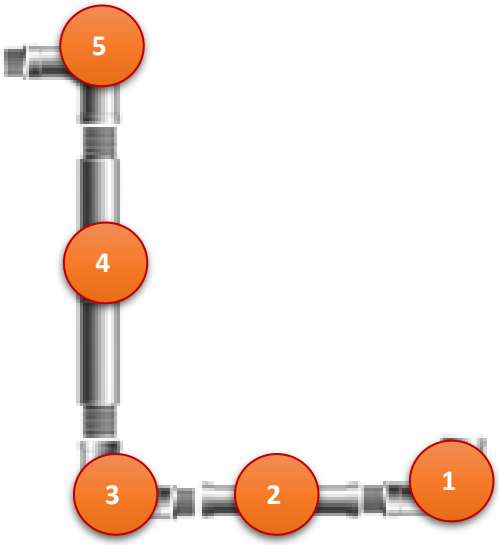


Figura 22. *Brazo metálico enumerado.*

Fuente: Elaboración propia

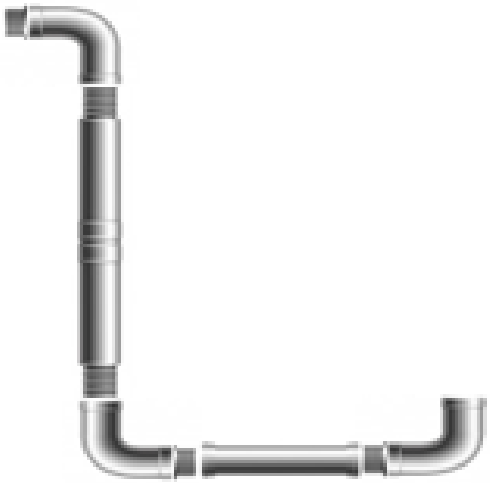


Figura 23. Brazo metálico armado. Fuente:

Elaboración propia

Anexo 4: Construcción del Stick

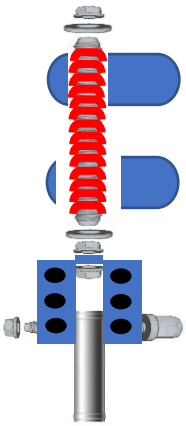


Figura 24. *Fabricación del Stick.*

Fuente: Elaboración propia

Anexo 5: Cámara para adquirir las imágenes

Figura 25. Cámara propia para adquirir imágenes.

Fuente: *Elaboración propia*

Anexo 6: Selección de los algoritmos

Tabla 8. *Algoritmos de clasificación*.

No.	Algorithm/class	1	2	3	4	5	6	7	8	9	Total
1	Global thresholding	4	2	1	1	1	1	1	1	1	13
2	Fractal analysis	4	3	2	1	1	1	1	2	1	16
3	Quaternion edge detection	2	1	3	3	1	1	1	4	1	17
4	FMC	1	1	3	3	3	1	1	1	4	18
5	SVM	3	4	3	2	2	1	1	1	1	18
6	Dynamic thresholding	3	4	3	2	2	1	1	1	2	19
7	Region splitting	3	3	4	2	2	1	1	2	1	19
8	Canny edge detection	3	4	3	1	2	1	2	2	1	19
9	Wavelet transform	3	1	3	4	3	1	1	1	2	19
10	Region merging/growing	3	3	4	2	2	1	1	2	2	20
11	Region merging and splitting	3	3	2	2	3	1	1	1	4	20
12	Clustering	2	3	3	4	1	1	1	3	2	20
13	Graph (MST)	3	3	3	3	1	1	1	1	4	20
14	Valley edge detection	2	3	1	2	1	1	4	4	3	21
15	Fractional differential	2	3	2	3	1	1	4	3	2	21
16	Multiple scales	3	3	2	3	1	1	3	4	1	21
17	Neural network	2	2	2	2	2	2	3	3	4	22
18	Deep learning	2	2	2	4	2	2	4	3	3	24

CNN

(Red Neuronal Convolucioal)

Nota: (Wang et al., 2019)

Anexo 7: Adquisición de las imágenes de las grietas de asfalto

Figura 26. *Procesar la imagen y aplicando thresholding binario.*

Fuente: Elaboración propia

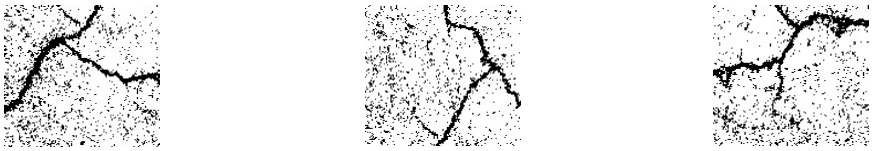


Figura 27. *Recortes de imágenes de 256x256 píxeles.*

Fuente: Elaboración propia

(a) (b)

Figura 28. *Imágenes sin grietas (a) y con grietas (b).*

Fuente: Elaboración propia

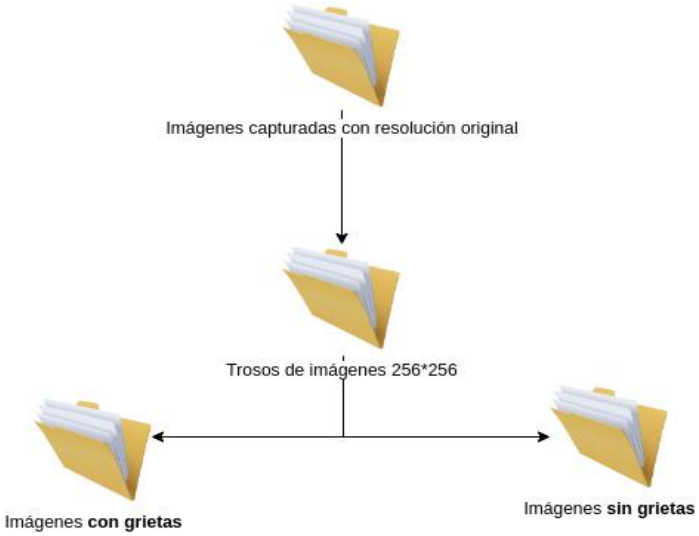
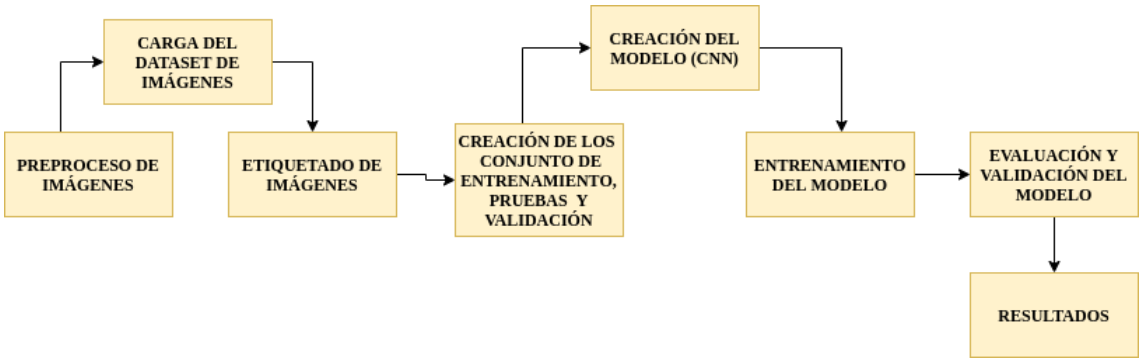


Figura 29. Creación del Dataset.

Fuente: *Elaboración propia*

Anexo 8: Implementación del algoritmo seleccionado

Tabla 9. Diagrama de bloques del método propuesto.



Nota: Elaboración propia

Anexo 9: Proceso de imágenes

Figura 30. Imágenes en memoria.

Fuente: *Elaboración propia*

```
def conversion_images(path_imgs, path_save):
    :param path_imgs: Dirección fisica de la carpeta de imagenes a tratar
    :param path_save: Dirección fisica de la carpeta donde se va a guardar
    list_names = get_list_names(path_imgs)  # Extrae la lista de nombres
    for name in list_names:  # recorre la lista de nombres
        path_image = f'{path_imgs}{name}'  # path fisico de cada imagen
        img_read = cv2.imread(path_image, 0)
        alpha = 1.1  # Control de contraste simple
        beta = 30  # Control de brillo simple
        img_read =  alpha* img_read + beta
        cv2.imwrite(f'{path_save}{name}', img_read)
        # global thresholding binary
        ret1, th1 = cv2.threshold(img_read, 120, 255, cv2.THRESH_BINARY)
        cv2.imwrite(f'{path_save}{name}' , th1)
        print(f'Imagen procesada {name} : {th1.shape}')
```

Fuente: *Elaboración propia*

Anexo 10: Carga del data set de imágenes

```
imagenes = []

directorios = []

cant_directorios = []

ruta_anterior = ''

cant = 0

tiempo_inicio = time()

# Funición para calcular el tiempo de ejecución la cual recibe dos parametros

# hora de fin y la hora de inicio calcula la diferencia y lo transforma en segundos y lo devuelve

print("leyendo imagenes de ", ruta_imagen)

for raiz, nombres_dir, nombres_archivos in os.walk(ruta_imagen):

    for nombre_archivo in nombres_archivos:

        if re.search("\.(jpg|jpeg|png|bmp|tiff)$", nombre_archivo):

            cant = cant + 1

            ruta_archivo = os.path.join(raiz, nombre_archivo)

            imagen = plt.imread(ruta_archivo)

            imagenes.append(imagen)

            b = "Leyendo..." + str(cant)

            print(b, end="\r")

            if ruta_anterior != raiz:

                print(raiz, cant)

                ruta_anterior = raiz

                directorios.append(raiz)
```

```
            cant_directorios.append(cant)
            cant = 0
cant_directorios.append(cant)


cant_directorios = cant_directorios[1:]
cant_directorios[0] = cant_directorios[0] + 1
print('Directorios leidos:', len(directorios))
print("Imagenes en cada directorio", cant_directorios)
print('Suma total de imagenes en subdirs:', sum(cant_directorios))
```

Fuente: *Elaboración propia*

Anexo 11: Etiquetado de imágenes

```
# Creamos las etiquetas

etiquetas = []

indice = 0

for cantidad in cant_directorios:

    for i in range(cantidad):

        etiquetas.append(indice)

    indice = indice + 1

print("Cantidad etiquetas creadas: ", len(etiquetas))

grietas = []

indice = 0

for directorio in directorios:

    nombre = directorio.split(os.sep)

    print(indice, nombre[len(nombre) - 1])

    grietas.append(nombre[len(nombre) - 1])

    indice = indice + 1

y = np.array(etiquetas)

X = np.array(imagenes, dtype=np.uint8)  # convierto de lista a numpy

# Encuentra los números únicos de las etiquetas

clases = np.unique(y)

cant_clases = len(clases)

print('Número total de salidas : ', cant_clases)

print('Clases de salida : ', clases)
```

Fuente: *Elaboración propia*

Anexo 12: Creación del conjunto de entramiento, pruebas y validación

Figura 34. Grupo de entrenamiento. Fuente: *Elaboración propia*

```
# Creamos Sets de Entrenamiento y Test

train_X, test_X, train_Y, test_Y = train_test_split(X, y, test_size=0.2)

print('Datos para entrenamiento : ', train_X.shape, train_Y.shape)

print('Datos para pruebas : ', test_X.shape, test_Y.shape)

# A contnuación ya pasamos a preprocesar las imagenes para el diseño de la red convolucional

train_X = train_X.reshape(train_X.shape[0], 256, 256, 1)

test_X = test_X.reshape(test_X.shape[0], 256, 256, 1)

train_X = train_X.astype('float32')

test_X = test_X.astype('float32')

train_X = train_X / 255.

test_X = test_X / 255.

# Hacemos el One-hot Encoding para la red

# Change the etiquetas from categorical to one-hot encoding

train_Y_one_hot = to_categorical(train_Y)

test_Y_one_hot = to_categorical(test_Y)
```

```
# Visualice el cambio para la etiqueta de categoría utilizando codificación de uno en caliente
print('Etiqueta original:', train_Y[0])
print('Después de la conversión a one-hot:', train_Y_one_hot[0])
# Creamos el Set de Entrenamiento y Validación
# Mezclar todo y crear los grupos de entrenamiento y testing
train_X, valid_X, train_label, valid_label = train_test_split(train_X,
                                                              train_Y_one_hot,
                                                              test_size=0.2,
                                                              random_state=13)
print(train_X.shape, valid_X.shape, train_label.shape, valid_label.shape)
```

Fuente: Elaboración propia

Anexo 13: Creación del modelo (CNN)

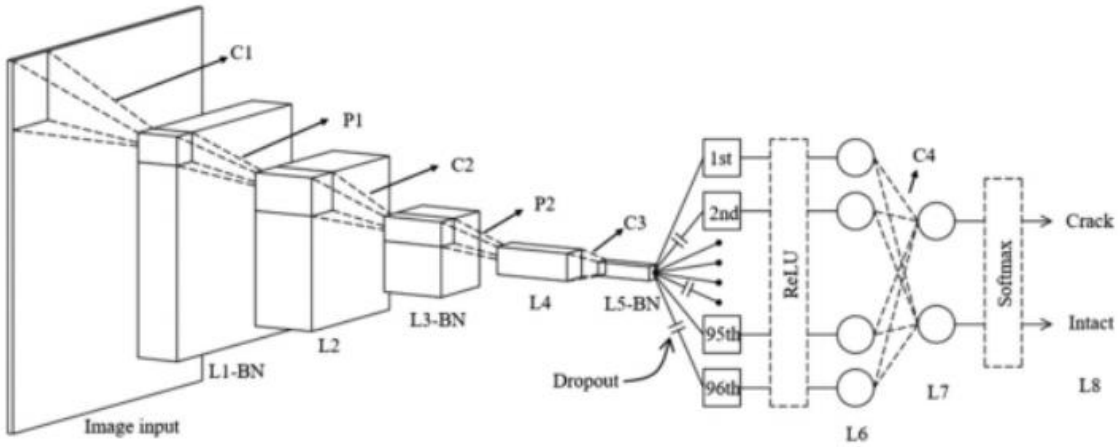


Figura 36. Arquitectura de la red neuronal convolucional.

Fuente: Young-Jin Cha, Wooram Choi

```
# Creamos el modelo de CNN

# declaramos variables con los parámetros de configuración de la red

INIT_LR = 1e-3  # Valor inicial de learning rate. El valor 1e-3 corresponde con 0.001

epochs = 100  # Cantidad de iteraciones completas al conjunto de imagenes de entrenamiento

batch_size = 100  # cantidad de imágenes que se toman a la vez en memoria

grietas_model = Sequential()

# priemra capa

grietas_model.add(Conv2D(24, kernel_size=(3, 3), input_shape=(256, 256, 1)))

grietas_model.add(MaxPooling2D((7, 7)))

# segunda capa

grietas_model.add(Conv2D(48, kernel_size=(3, 3)))

grietas_model.add(MaxPooling2D((4, 4)))

# tercera capa
```

```
grietas_model.add(Conv2D(96, kernel_size=(3, 3)))
grietas_model.add(Activation('relu'))
# cuarta capa
grietas_model.add(Conv2D(2, kernel_size=(3, 3), input_shape=(1, 1)))
grietas_model.add(Flatten())
grietas_model.add(Dense(2, activation='softmax'))
grietas_model.summary()
grietas_model.compile(loss=keras.losses.categorical_crossentropy,
                      optimizer=keras.optimizers.Adagrad(lr=INIT_LR,
                                                         decay=INIT_LR / 100), metrics=['accuracy'])
```

Fuente: Elaboración propia

Anexo 14: Evaluación

```
Y_pred = grietas_model.predict_classes(test_X)

df = pd.DataFrame(Y_pred)

print(df)

print('Matriz de confusión')

matrix = confusion_matrix(test_Y, Y_pred)

df = pd.DataFrame(matrix)

df.index = ['con_getas', 'sin_grietas']

df.columns = ['con_gririetas', 'sin_grietas']

print(df)

#Calculo de indicadores

tn, fp, fn, tp = matrix.ravel()

print((tn, fp, fn, tp))

exactitud = (tp + tn) / (tn + fp + fn + tp)

sensibilidad = tp / (tp + fn)

especificidad = tn / (tn + fp)

auc = 0.5 * (exactitud + sensibilidad)

error = 1 - exactitud

tiempo_transcurrido = time() - tiempo_inicio

print(

    f' Exactitud: {exactitud:.2f}\n '

    f'Sensibilidad:{sensibilidad:.2f}\n '

    f'Especificidad: {especificidad:.2f}\n '

    f'AUC: {auc:.2f}\n '
```

```
f'Error: {error:.2f}\n '
f'Tiempo: {tiempo_transcurrido:.2f}')
```

Fuente: *Elaboración propia*

Anexo 15: Validación

```
# Model configuration
nro_epochs = 100  # Cantidad de iteraciones completas al conjunto de imagenes de entrenamiento
batch_size = 50  # cantidad de imágenes que se toman a la vez en memoria
validation_split = 0.2
verbosity = 1
num_folds = 10
# Definir los contenedores de puntuación por fold
acc_per_fold = []
loss_per_fold = []
# Fusionar las entradas y los objetivos
inputs = np.concatenate((train_X, valid_X), axis=0)
targets = np.concatenate((train_label, valid_label), axis=0)
# Definir el validador de K-fold Cross Validator
kfold = KFold(n_splits=num_folds, shuffle=True)
# Evaluación del modelo de validación cruzada de K-fold
fold_no = 1
for train, test in kfold.split(inputs, targets):
  # Generar una impresión
```

```
    print(f'Entrenamiento para fold {fold_no} ...')
    grietas_model = creacion_modelo()
    # Ajustar los datos al modelo
    history = grietas_model.fit(inputs[train], targets[train],
                                batch_size=batch_size,
                                epochs=nro_epochs,
                                verbose=verbosity,
                                validation_split=validation_split)
    # Generar métricas de generalización
    scores = grietas_model.evaluate(inputs[test], targets[test], verbose=0)
    print(
        f'Puntuación por fold {fold_no}: {grietas_model.metrics_names[0]} de {scores[0]}; '
        f'{grietas_model.metrics_names[1]} hasta {scores[1] * 100}%')
    acc_per_fold.append(scores[1] * 100)
    loss_per_fold.append(scores[0])
    grietas_model.save(f'model_binary_{fold_no}.h5py')
    # Aumentar el número de folds
    fold_no = fold_no + 1
# == Proporcionar las puntuaciones medias ==
for i in range(0, len(acc_per_fold)):
    print(f'> Fold {i + 1} - Pérdida: {loss_per_fold[i]} - Exactitud: {acc_per_fold[i]}%')
print('Puntuaciones para todos los folds:')
```

```
print(f'> Exactitud: {np.mean(acc_per_fold, dtype=np.float32)} (+- {np.std(acc_per_fold, dtype=np.float32)})')

print(f'> Pérdida: {np.mean(loss_per_fold)}')
```

Fuente: Elaboración propia

Anexo 16: Resumen de los resultado obtenidos

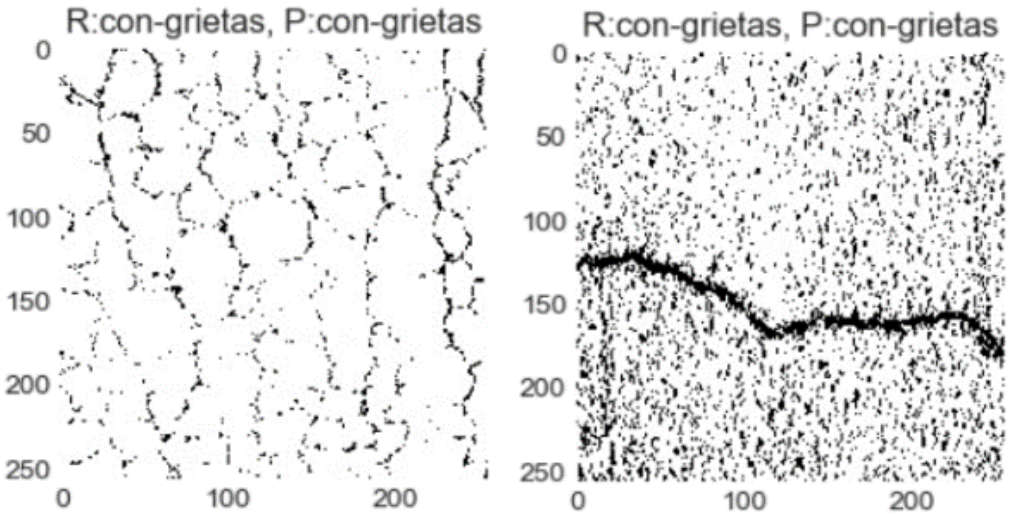


Figura 41. Pruebas y resultados.

Fuente: Elaboración propia

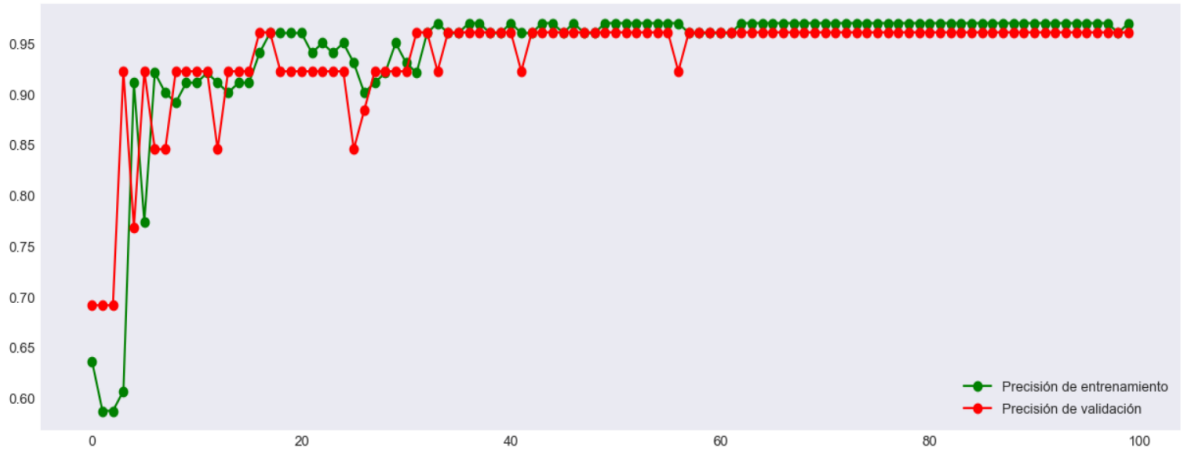


Figura 42. Iteraciones de entrenamiento.

Fuente: *Elaboración propia*

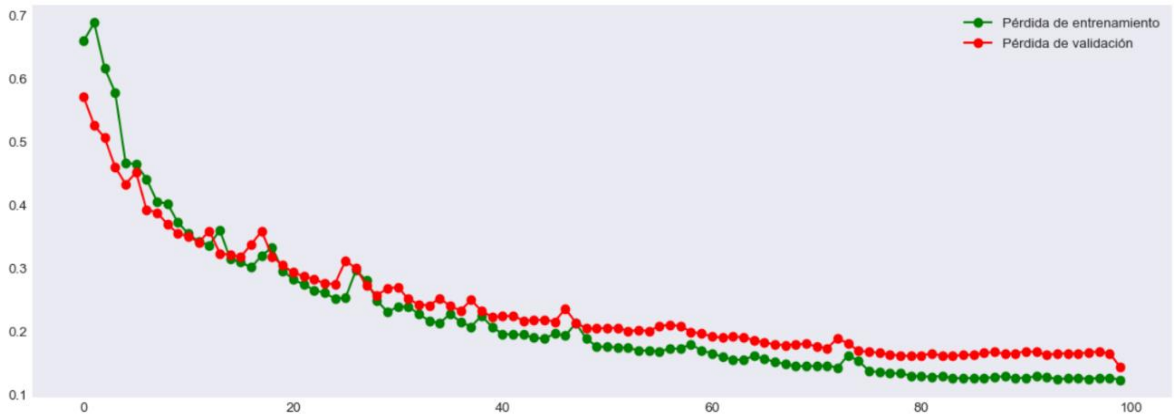


Figura 43. Iteraciones de pérdida.

Fuente: *Elaboración propia*

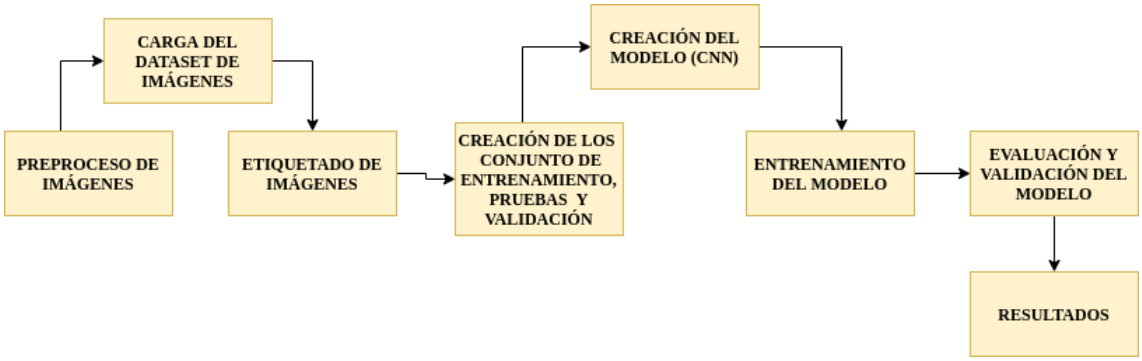
Anexo 17: Manual de usuario

Introducción

En este apartado se pretende orientar a los interesados a utilizar la tecnología de procesamiento de imágenes para identificar grietas en el asfalto, aportando la línea de investigación definida por la escuela como ciencia de la computación. A continuación, se describe las funcionalidades principales representadas en bloques figura:

Tabla 9.

Descripción gráfica del diagrama de los de los principales procesos de la técnica planteada



Nota: (Elaboración propia)

Mapa del sistema

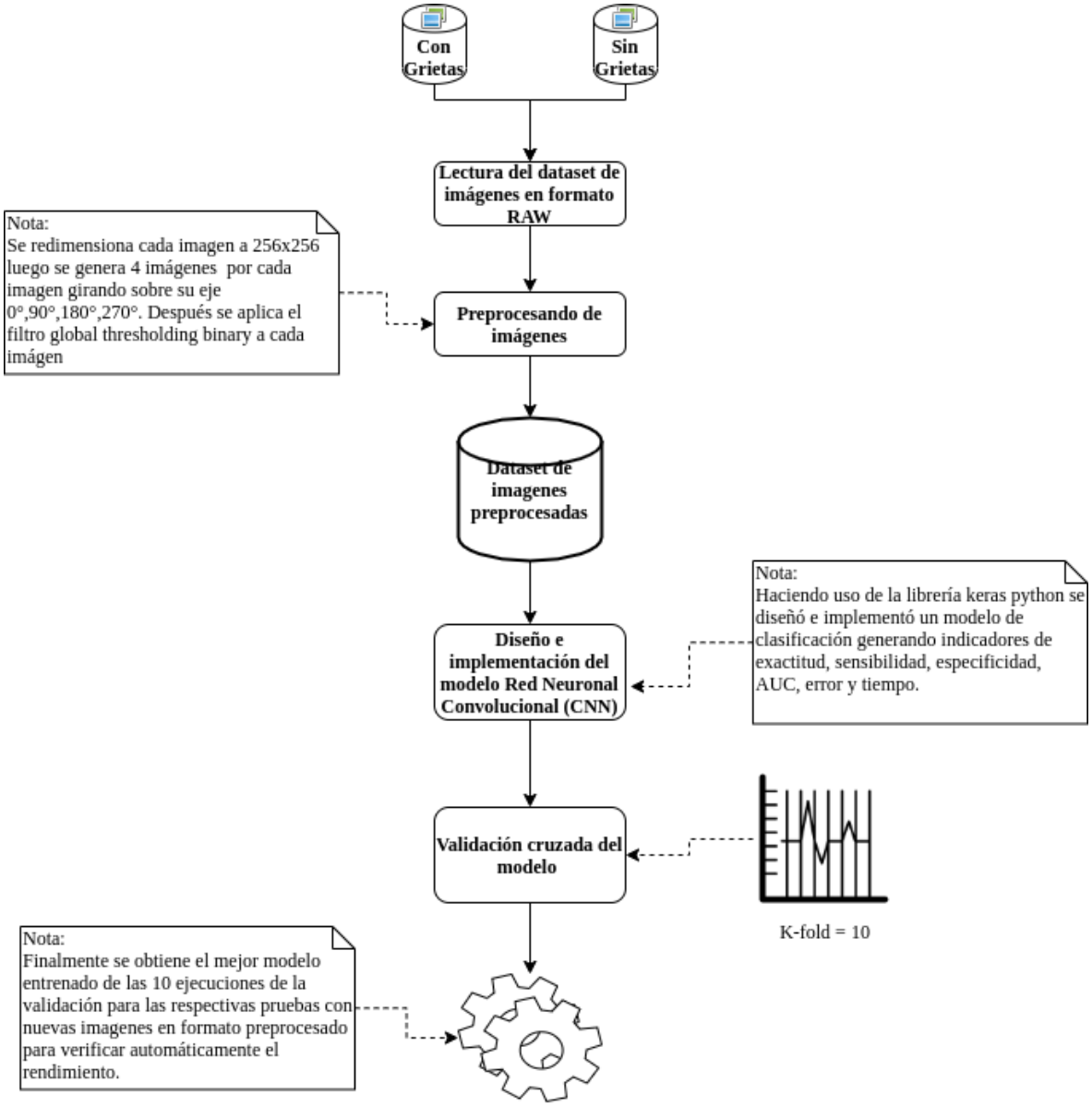


Figura 37. *Modelo Lógico*.

Fuente: (Elaboración propia)

Navegación

El método propuesto es este proyecto consta con las siguientes carpetas de navegación:

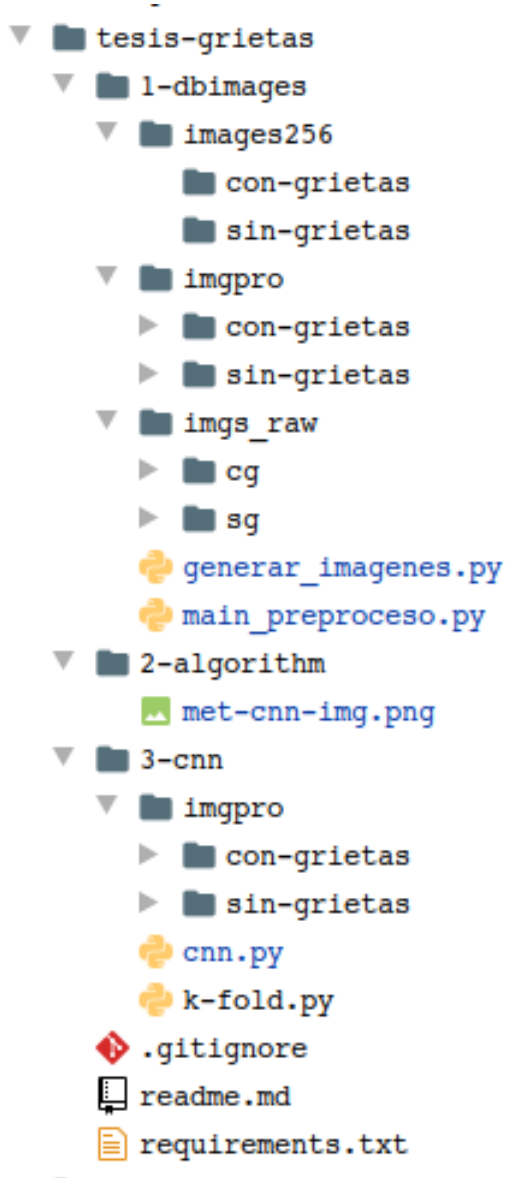


Figura 38. Estructura de carpetas del proyecto.

Fuente: *(Elaboración propia)*

Tabla 9.

Descripción de la carpeta de navegación

Carpeta	Descripción
robus-grietas/	Carpeta principal del proyecto
1-dbimagenes/	Carpeta para el Pre-proceso de imágenes
images256/	Carpeta que contiene las imágenes redimensionadas a 256x256
con-grietas/	Carpeta que contiene las imágenes redimensionadas con grietas
sin-grietas/	Carpeta que contiene las imágenes redimensionadas sin grietas
imgpro/	Carpeta que contiene las imágenes procesadas para entrenamiento de la red neuronal artificial CNN
con-grietas/	Carpeta que contiene las imágenes procesadas con grietas
sin-grietas/	Carpeta que contiene las imágenes procesadas sin grietas
imgs_raw/	Carpeta que contiene las imágenes en formato crudo tal cual son captadas por el lente de la cámara.
cg/	Carpeta que contiene las imágenes con grietas
sg/	Carpeta que contiene las imágenes sin grietas
generar_imagenes.py	Archivo python para poder preprocesar las imágenes y generar nuevas imágenes por cada ángulo 0°,90°,180°,270°.
main_preproceso.py	Archivo python para poder preprocesar las imágenes generadas añadiendo el filtro global thresholding binary a cada una y dejando listo para ser utilizadas para el entrenamiento de la Red Neuronal Convolucional (CNN)
2-algorithm/met-cnn-img.png	Es una imagen del modelo de capas utilizado para modelar Red Neuronal Convolucional (CNN)
3-cnn/imgpro	Dataset de imágenes generadas en [1-dbimagenes]
cnn.py	El archivo principal python que carga el dataset divide los conjuntos de entrenamiento 80% y pruebas 20%, etiqueta las clases objetivo (0=con grietas, 1=sin grietas), genera el modelo, compila, entrena y evalúa los resultados. Además, se genera la matriz de confusión extrayendo los indicadores propuestos (exactitud, sensibilidad, especificidad, AUC, Error y Tiempo de entrenamiento y pruebas)
k-fold.py	Archivo python para evaluar el modelo propuesto utilizando la validación cruzada k-fold utilizando configurando el valor constante k=10
requirements.txt	El archivo requirements.txt se usa para especificar los paquetes de software de Python requerido para ejecutar el proyecto. Por lo general el archivo txt se encuentra en el directorio raíz del proyecto.

Nota: Elaboración propia

Instalar la aplicación

Para poder instalar la aplicación es necesario instalar un en IDE de desarrollo Python o algún editor de código con acceso al terminal. A continuación, se muestra los pasos que se han tenido en cuenta para esta propuesta.

Instalar Python3

Se sugiere descargar la versión proporcionada en el sitio web oficial ver Figura 3. Se debe seguir los pasos para agregar Python a la RUTA en la primera ventana del instalador. Por supuesto, debe desinstalar cualquier otra versión que se tenga, porque si detecta más de una versión a la vez, no funcionará correctamente.

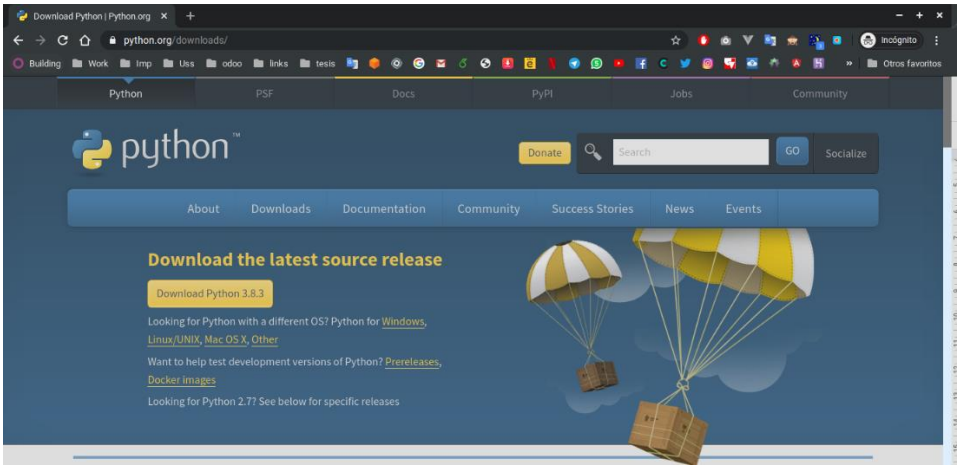


Figura 39. Descargar python3

Fuente: *https://www.python.org/downloads/*

Instalar el IDE PyCharm

PyCharm es un IDE desarrollo completo que si bien es cierto es pagado se puede obtener una versión de prueba profesional utilizando un correo institucional con extensión .edu para ello se debe acceder a la página principal Figura 47.

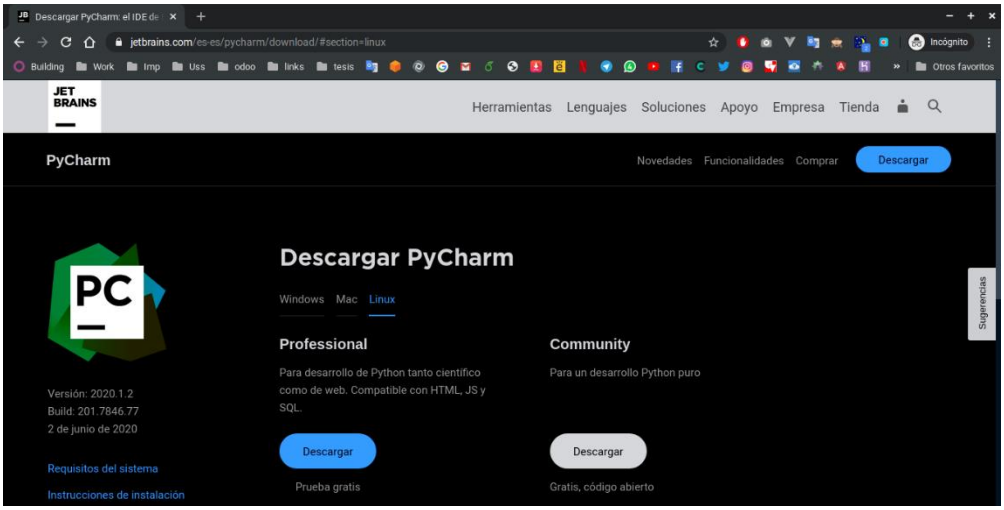


Figura 40. Descargar el IDE PyCharm.

Fuente: *Elaboración propia*

Después de instalar y ejecutar el IDE se procede abrir la carpeta que contenga la aplicación de tesis propuesta llamada tesis-grietas descrita anteriormente en la Tabla 10.

Configurar un entorno virtual

PyCharm hace posible utilizar la herramienta virtualenv (venv) para crear un entorno virtual aislado específico del proyecto. El objetivo principal de los entornos virtuales es administrar la configuración y las dependencias de un proyecto en particular, independientemente de otros proyectos de Python. La herramienta virtualenv viene incluida con PyCharm, por lo que el usuario no necesita instalarla.

Crear un entorno virtual

1. Asegúrese de haber descargado e instalado Python en su computadora.
2. Presione Ctrl+Alt+S para abrir la Configuración / Preferencias del proyecto
3. En el cuadro de diálogo Configuración / Preferencias Ctrl+Alt+S , seleccione Proyecto <nombre del proyecto> / Intérprete de proyectos. Haga clic en el icono y seleccione Agregar.
4. Agregar intérprete de Python, seleccione Virtualenv Environment. Las siguientes acciones dependen de si el entorno virtual existía antes.

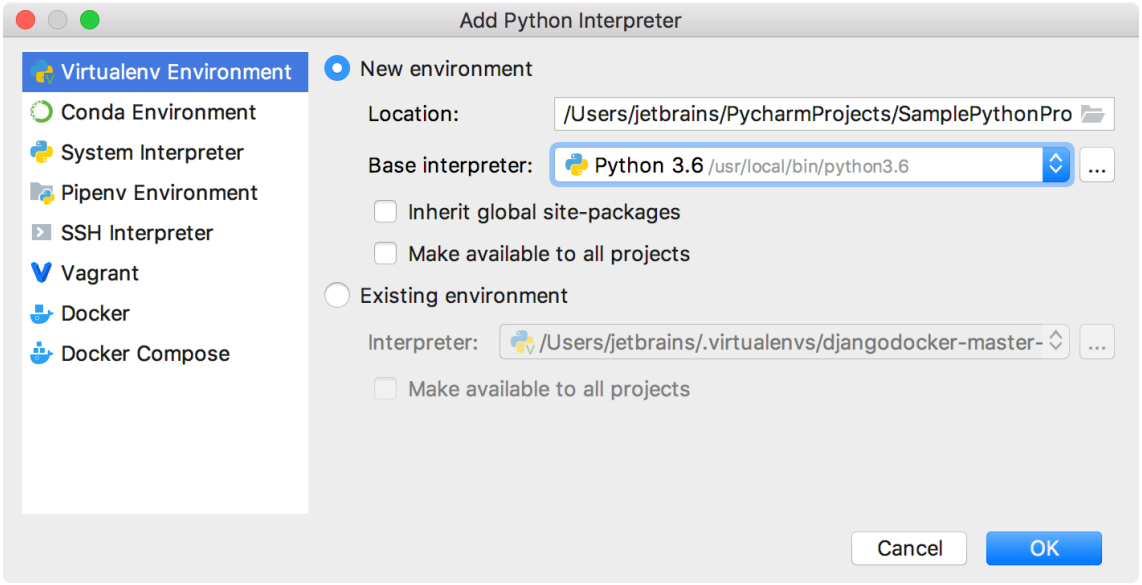


Figura 41. *Configurando entorno. Fuente:* (elaboración propia)

Si se selecciona Nuevo entorno:

1. Especifique la ubicación del nuevo entorno virtual en el campo de texto, o haga clic y busque la ubicación en su sistema de archivos. Tenga en cuenta que el directorio donde debe ubicarse el nuevo entorno virtual debe estar vacío.
2. Elija el intérprete base de la lista o haga clic y busque un ejecutable de Python en su sistema de archivos.

   Si PyCharm no detecta Python en su máquina, proporciona dos opciones: descargar las últimas versiones de Python desde python.org o para especificar una ruta al ejecutable de Python (en caso de instalación no estándar).

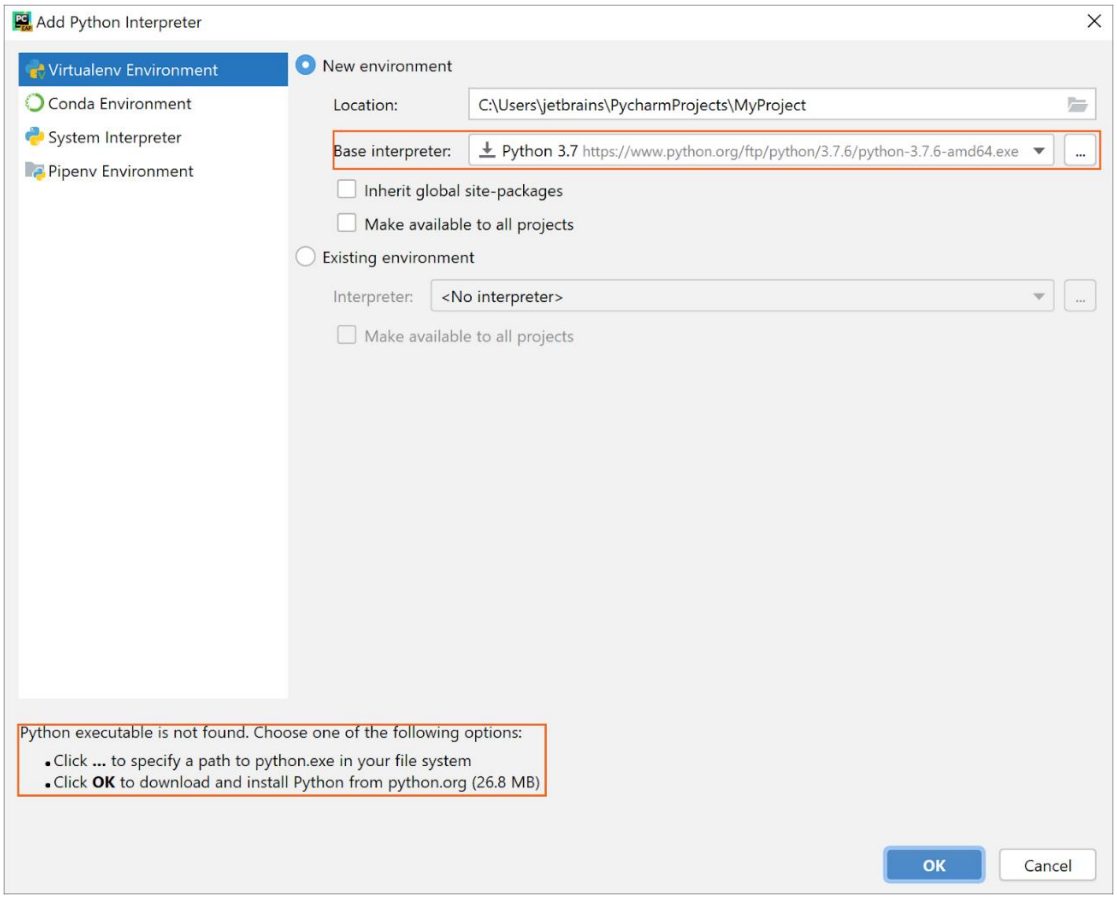


Figura 42. *Configurando directorio (elaboración).*

Fuente: Elaboración propia

3. Seleccione la casilla de verificación Heredar paquetes globales del sitio si desea heredar su directorio global de paquetes del sitio. Esta casilla de verificación corresponde a la -system-site-packages opción de la herramienta virtualenv.
4. Seleccione la casilla de verificación Poner a disposición de todos los proyectos, si es necesario.

Si se selecciona el entorno existente:

a. Expanda la lista de Intérpretes y seleccione cualquiera de los intérpretes existentes. Alternativamente, haga clic y especifique una ruta al ejecutable de Python en su sistema de archivos, por ejemplo, C: \ Python36 \ python.exe.
b. Seleccione la casilla de verificación Poner a disposición de todos los proyectos si es necesario.

5. Haga clic en Aceptar para completar la tarea.

Instalar paquetes en el entorno virtual

Para instalar los paquetes necesarios para poder ejecutar la propuesta se debe abrir el terminal integrado del IDE esto activar automáticamente el entorno virtual creado anteriormente luego se debe ejecutar el comando pip3 install requirements.txt esto instalará todas las dependencias en el entorno y el proyecto quedará listo para ser ejecutado.

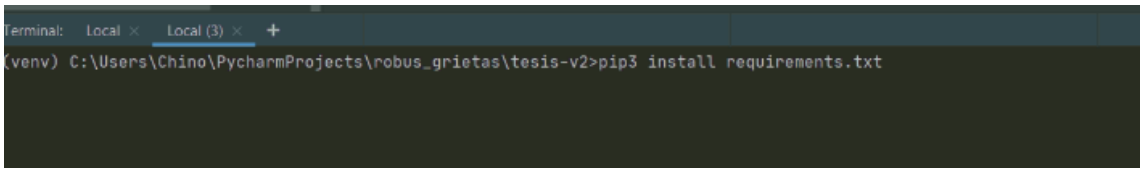


Figura 43. *Configurando terminal.*

Fuente: (Elaboración propia)

Anexo 18: Interfaz del algoritmo red neuronal convolucional (CNN)

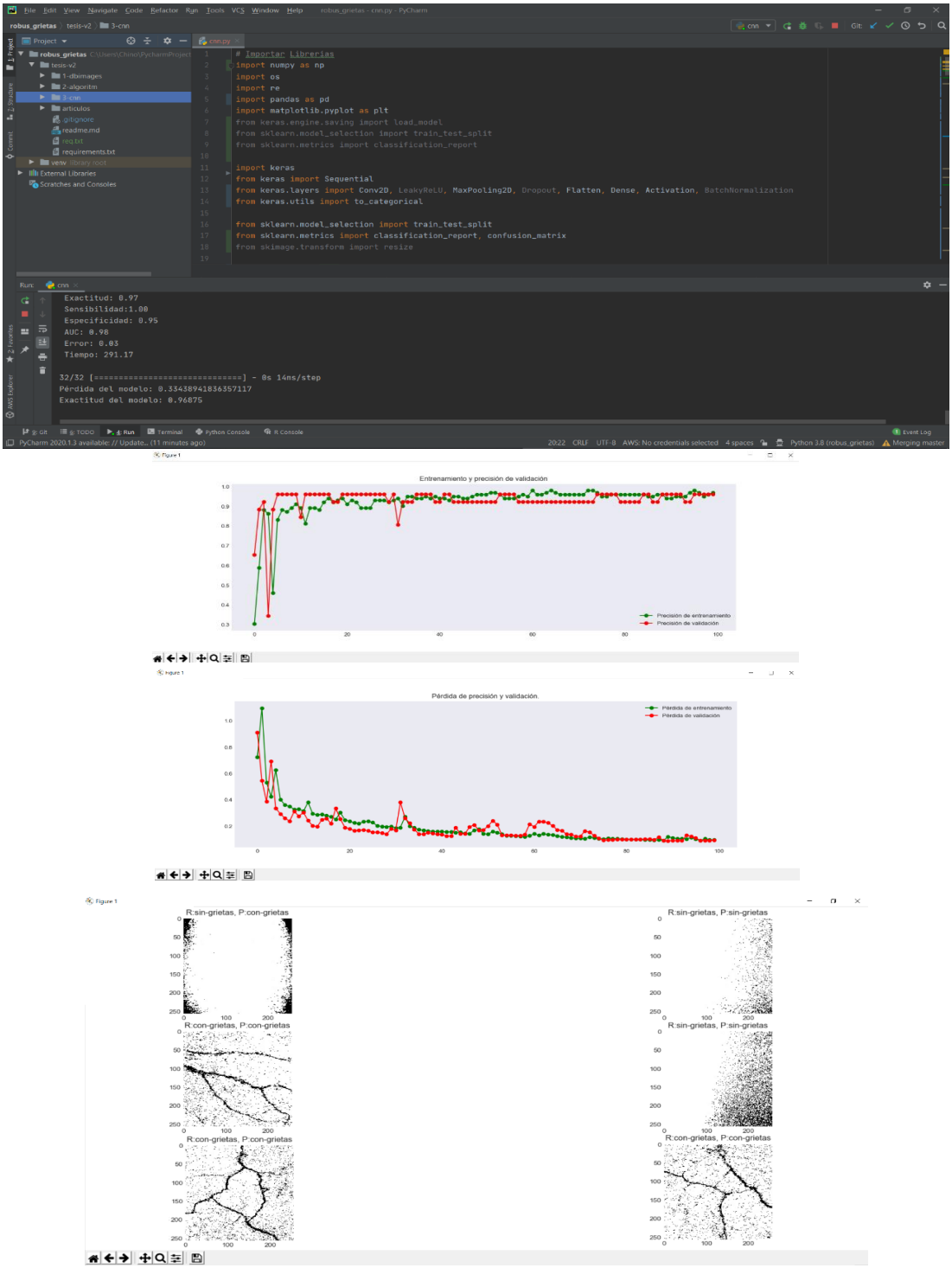


Figura 44. *Interfaz CNN – Python.*

Fuente: Elaboración Propia.

Anexo 19: Detalle de la implementación del algoritmo red neuronal convolucional (CNN)

```
# Importar Librerías

import numpy as np

import os

import re

import matplotlib.pyplot as plt

from keras.engine.saving import load_model

from sklearn.model_selection import train_test_split

from sklearn.metrics import classification_report

import keras

from keras import Sequential

from keras.layers import Conv2D, LeakyReLU, MaxPooling2D, Dropout, Flatten, Dense, Activation, BatchNormalization

from keras.utils import to_categorical

from sklearn.model_selection import train_test_split

from sklearn.model_selection import KFold

#Cargar Imágenes

clases = ["SIN GRIETAS", "CON GRIETAS"]

dirname = os.path.join(os.getcwd(), 'imgpro')

imgpath = dirname + os.sep

images = []

directories = []

dircount = []

prevRoot = ''
```

```
cant = 0
print("leyendo imagenes de ", imgpath)
for root, dirnames, filenames in os.walk(imgpath):
    for filename in filenames:
        if re.search("\.(jpg|jpeg|png|bmp|tiff)$", filename):
            cant = cant + 1
            filepath = os.path.join(root, filename)
            image = plt.imread(filepath)
            images.append(image)
            b = "Leyendo..." + str(cant)
            print(b, end="\r")
            if prevRoot != root:
                print(root, cant)
                prevRoot = root
                directories.append(root)
                dircount.append(cant)
                cant = 0
dircount.append(cant)
dircount = dircount[1:]
dircount[0] = dircount[0] + 1
print('Directorios leidos:', len(directories))
print("Imagenes en cada directorio", dircount)
print('Suma total de imagenes en subdirs:', sum(dircount))
```

Fuente: Elaboración Propia.

```
# Creamos las etiquetas
labels = []
indice = 0
for cantidad in dircount:
    for i in range(cantidad):
        labels.append(indice)
    indice = indice + 1
print("Cantidad etiquetas creadas: ", len(labels))
grietas = []
indice = 0
for directorio in directories:
    name = directorio.split(os.sep)
    print(indice, name[len(name) - 1])
    grietas.append(name[len(name) - 1])
    indice = indice + 1
y = np.array(labels)
X = np.array(images, dtype=np.uint8)  # convierto de lista a numpy
# Encuentra los números únicos de las etiquetas
classes = np.unique(y)
nClasses = len(classes)
print('Número total de salidas : ', nClasses)
print('Clases de salida : ', classes)
```

Fuente: Elaboración Propia

```
# Creamos Sets de Entrenamiento y Test
train_X, test_X, train_Y, test_Y = train_test_split(X, y, test_size=0.2)
print('Datos para entrenamiento : ', train_X.shape, train_Y.shape)
print('Datos para pruebas : ', test_X.shape, test_Y.shape)

# A contnuación ya pasamos a preprocesar las images256 para el diseño de la red convolucional
train_X = train_X.reshape(train_X.shape[0], 256, 256, 1)
test_X = test_X.reshape(test_X.shape[0], 256, 256, 1)

train_X = train_X.astype('float32')
test_X = test_X.astype('float32')
train_X = train_X / 255.
test_X = test_X / 255.
#
# Hacemos el One-hot Encoding para la red
# Change the labels from categorical to one-hot encoding
train_Y_one_hot = to_categorical(train_Y)
test_Y_one_hot = to_categorical(test_Y)

# Visualice el cambio para la etiqueta de categoría utilizando codificación de uno en caliente
print('Etiqueta original:', train_Y[0])
print('Después de la conversión a one-hot:', train_Y_one_hot[0])
```

Fuente: Elaboración Propia

```
# Creamos el Set de Entrenamiento y Validación
# Mezclar todo y crear los grupos de entrenamiento y testing
train_X, valid_X, train_label, valid_label = train_test_split(train_X,
                                                              train_Y_one_hot,
                                                              test_size=0.2,
                                                              random_state=13)
print(train_X.shape, valid_X.shape, train_label.shape, valid_label.shape)
def creacion_modelo():
```

Fuente: Elaboración Propia