UNIVERSIDAD NACIONAL DE INGENIERÍA

UNIDAD DE POSGRADO
FACULTAD DE INGENIERÍA CIVIL

TESIS

"DETECCIÓN AUTOMÁTICA DEL ESTADO DE SUPERFICIES DE PAVIMENTO FLEXIBLE EN LA CIUDAD DE PIURA UTILIZANDO APRENDIZAJE PROFUNDO"

PARA OBTENER EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS EN INGENIERÍA CIVIL CON MENCIÓN EN TRANSPORTES

ELABORADA POR:

HILDER JAVIER JIMENEZ PESANTES

ASESOR:

Dr. ALAN FISCHER AYALA OBREGÓN

LIMA, PERÚ

2024

“DETECCIÓN AUTOMÁTICA DEL ESTADO DE SUPERFICIES DE PAVIMENTO FLEXIBLE EN LA CIUDAD DE PIURA UTILIZANDO APRENDIZAJE PROFUNDO”

HILDER JAVIER JIMENEZ PESANTES

Presentado a la Unidad de Posgrado de la Facultad de Ingeniería Civil en cumplimiento parcial de los requerimientos para el grado académico de:

MAESTRO EN CIENCIAS EN INGENIERÍA CIVIL CON MENCIÓN EN TRANSPORTES

DE LA

UNIVERSIDAD NACIONAL DE INGENIERÍA

2024

Autor	: Ing. Hilder Javier Jimenez Pesantes
Recomendado	: Dr. Alan Fischer Ayala Obregón Asesor de la Tesis
Aceptado por	: Dra. Heddy Marcela Jimenez Yabar Directora (e) de la Unidad de Posgrado

@ 2024; Universidad Nacional de Ingeniería, todos los derechos reservados o el autor autoriza a la UNI-FIC a reproducir la tesis en su totalidad o en partes.

DEDICATORIA

Dedicó el presente trabajo de investigación a Dios.

A mi padre y madre por permitirme innovar desde casa.

A mis hermanas y a mi familia por el gran amor que me brindan.

A mis compañeros de FIC-UNP y ahora colegas.

Así también a mis amigos quienes me han brindado su apoyo y cada día me enseñan que hay que trabajar por un propósito, no por un aplauso.

AGRADECIMIENTOS

A la Universidad Nacional de Ingeniería y a sus docentes por las cátedras impartidas con ética profesional y calidad extraordinaria. Permitiéndome prosperar y culminar de gran manera la Maestría en Ciencias en Ingeniería Civil con Mención en Transportes.

Mi más grande gratitud a mi asesor Dr. Alan Fischer Ayala Obregón por haberme brindado su apoyo inestimable compartiendo sus conocimientos y experiencia para el desarrollo de esta investigación.

A los especialistas M.Sc. Ing. Leonardo Flores González y M.Sc. Ing. Fanny Beatriz Eto Chero por sus observaciones y sugerencias, las cuales han aportado enormemente en este trabajo de investigación.

A mi familia por su apoyo incondicional en cada etapa de mi vida profesional.

He de mostrar mi gratitud a mis compañeros y amigos quienes de forma directa e indirecta ayudaron en el desarrollo de esta tesis de maestría.

ÍNDICE DE CONTENIDOS:

Pág.

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

La presente tesis se centró en mejorar la precisión en el proceso de detección automática del estado de las superficies de pavimento flexible en la ciudad de Piura mediante el uso de aprendizaje profundo. El objetivo general fue determinar un modelo de aprendizaje profundo que lograra dicha mejora en la precisión.

Para ello, se entrenó y validó un modelo de aprendizaje profundo en comparación con el software *Pavimenta2* del Banco Interamericano de Desarrollo (*BID*). El modelo, desarrollado en *Python*, se validó experimentalmente hasta alcanzar el nivel de precisión deseado. Cabe destacar que las fotografías y videos de los pavimentos de Piura utilizados en el procesamiento fueron recopilados a través de cámaras de smartphone. Posteriormente, los resultados se validaron con Pavimenta2, el cual se ha incorporado de manera novedosa en este estudio.

La metodología de investigación empleada es de enfoque cuantitativo, con variables definidas operacionalmente. En este sentido, se elaboraron modelos que fueron analizados con base en los resultados obtenidos. Si bien el diseño es experimental, en el marco de control del estudio se clasifica como preexperimental. Debido a que se utilizó un grupo de imágenes por tipo de falla para el entrenamiento del modelo, sin contar con un grupo de control.

El trabajo se basó en utilizar cuatro redes neuronales de base y transferir su inteligencia para seleccionar el modelo superior en capacidad de predicción de grietas en el pavimento. Mediante técnicas de aprendizaje profundo, se alcanzó una precisión del 92.76 %, considerada excelente para fines de detección de imágenes.

Los experimentos demostraron que el modelo de inteligencia artificial desarrollado resultó eficaz en la detección de fallas en los pavimentos flexibles de Piura, posicionándose como una herramienta poderosa para asistir a los equipos de supervisión y mantenimiento en la programación de reparaciones y emisión de alertas tempranas de daños potenciales.

ABSTRACT

This thesis focused on improving the precision in the automatic detection process of the condition of flexible pavement surfaces in the city of Piura through the use of deep learning. The general objective was to determine a deep learning model that could achieve such improvement in precision.

To this end, a deep learning model was trained and validated in comparison with the Pavimenta2 software from the Inter-American Development Bank (*IDB*). The model, developed in Python, was experimentally validated until the desired level of precision was achieved. It is important to note that the photographs and videos of the Piura pavements used in the processing were collected through smartphone cameras. Subsequently, the results were validated with Pavimenta2, which has been incorporated in a novel way in this study.

The research methodology employed is a quantitative approach, with operationally defined variables. In this sense, models were developed and analyzed based on the obtained results. Although the design is experimental, in the control framework of the study, it is classified as pre-experimental due to the use of a group of images by type of failure for model training, without a control group.

The work was based on using four base neural networks and transferring their intelligence to select the superior model in the ability to predict cracks in the pavement. Through deep learning techniques, a precision of 92.76% was achieved, considered excellent for image detection purposes.

The experiments demonstrated that the developed artificial intelligence model was effective in detecting failures in the flexible pavements of Piura, positioning it as a powerful tool to assist the supervision and maintenance teams in scheduling repairs and issuing early warnings of potential damage.

INTRODUCCIÓN

El estudio de los avances recientes en visión por computadora, utilizando la inteligencia artificial y, específicamente, el aprendizaje profundo, está impulsando un método automático de detección de grietas en las redes viales de las ciudades.

En el Perú, este es aún un tema poco explorado. Por lo tanto, se busca enriquecer los conceptos y exhibir un modelo de detección de grietas que sirva de guía para futuras investigaciones. En este sentido, el objetivo de la tesis es determinar un modelo de aprendizaje profundo que permita mejorar la precisión en el proceso de detección automática del estado de las superficies de pavimento flexible en la ciudad de Piura.

El primer capítulo, "Protocolo de la investigación", desarrolla el problema de estudio; además, plantea los problemas, objetivos e hipótesis de manera general y específica, presenta las variables de la investigación y describe la metodología que guiará el desarrollo de la tesis.

En el segundo capítulo, se expone el marco teórico asociado con el tema de investigación. Esto comprende principalmente los conceptos de fallas en pavimento flexible y los avances recientes en visión por computadora, específicamente en el marco de la detección de fallas basadas en la técnica de aprendizaje profundo, la cual es un subconjunto del aprendizaje automático y del campo de la inteligencia artificial.

El tercer capítulo describe detalladamente el desarrollo del trabajo de investigación. Se detallan las plataformas y el hardware utilizados para entrenar las redes convolucionales, así como el ecosistema computacional empleado para los experimentos. Posteriormente, se presentan los resultados experimentales obtenidos utilizando tarjetas gráficas de alto rendimiento, y se analiza la precisión del modelo desarrollado, comparándolo con los resultados del software Pavimenta2 del Banco Interamericano de Desarrollo (BID).

Finalmente, se presentarán las conclusiones que responden a los objetivos de la tesis, así como las recomendaciones respecto a la detección de grietas y el mantenimiento de los pavimentos flexibles evaluados.

CAPÍTULO I: PROTOCOLO DE LA INVESTIGACIÓN

1.1 Identificación y Descripción del Problema de Estudio

Uno de los principales activos de las ciudades en el Perú y en el mundo son las vías urbanas. Estas son esenciales para el desarrollo de la economía de una ciudad; por tanto, tener redes viales de buena calidad es una necesidad para casi todas las actividades socioeconómicas. Por consiguiente, es de vital importancia monitorear el deterioro tanto por el paso del tiempo como por su uso, más aún cuando extensas partes del pavimento se degradan hasta llegar a un estado pésimo, entorpeciendo la transitabilidad vial y serviciabilidad que deben brindar.

Al buscar soluciones entre las tecnologías empleadas para la evaluación del estado del pavimento, se viene promoviendo el uso de la inteligencia artificial. En este caso, buscamos contribuir con mejorar el proceso de detección automática del estado de las superficies de pavimento flexible en la ciudad de Piura, de forma que sea eficiente en precisión.

1.1.1 Formulación del Problema General

La conservación de pavimentos en redes viales es primordial para garantizar la seguridad y comodidad de los usuarios, además de tener importantes beneficios económicos. A pesar de ello, según Rodríguez (2020) afirma que “existen miles de kilómetros que necesitan ser inspeccionados, y esta labor se realiza en la mayoría de los casos de forma manual mediante la inspección visual supervisada por expertos, siendo una tarea ineficiente en el tiempo” (p. XV).

Ahora bien, si nos ubicamos en el escenario del mundo real, cuando los gestores de vías urbanas de un organismo rector de infraestructura necesitan reparar dicho daño, necesitan información precisa y actualizada sobre los defectos del pavimento para tomar una decisión efectiva y gestionar de manera eficaz la red vial. Mediante los métodos tradicionales, como lo señalan Nguyen et al. (2016), “los inspectores humanos obtienen manualmente la información sobre los defectos de las carreteras. Pero estos métodos manuales son muy lentos e incómodos para los inspectores y usuarios de la vía” (p. 40).

Consecuentemente, un mayor gasto en la conservación de la vía hace más difícil su financiamiento, ocasionando una dilatación del tiempo y comúnmente se le presta atención ya cuando el pavimento llega a niveles intransitables. Esto va en aumento como lo muestra la Memoria Anual del año 2019, que indica que desde el 2001 al 2019 el gasto del

mantenimiento vial ha tenido un crecimiento anual promedio del 12 % (Provias Nacional, 2020).

A partir de lo expuesto, en esta investigación buscamos realizar una plataforma integral para la evaluación, monitoreo y mantenimiento de vías pavimentadas de manera más eficiente en tiempo y con resultados confiables. Para ello, se utilizará el método de Aprendizaje Profundo (*Deep Learning)*, una rama prometedora del aprendizaje automático (*Machine Learning)*. Como caso de estudio tomamos la ciudad de Piura, una de las más importantes de nuestra nación.

Al concluir este trabajo de investigación, se brindará información precisa y actualizada que facilitará la elaboración de un plan de mantenimiento en las vías urbanas de pavimento flexible, permitiendo así una gestión eficaz para prolongar su conservación en el tiempo. La solución o herramienta de software desarrollada en el marco de esta tesis se aplicará al estudio de otras vías de interés nacional mediante la técnica conocida como transferencia de aprendizaje.

Al formular el problema general, se busca orientar la investigación hacia la solución de un problema práctico y relevante, utilizando el Aprendizaje Profundo como herramienta para mejorar la precisión de la detección automática del estado del pavimento flexible en la ciudad de Piura. Por lo cual, el problema general de investigación como pregunta vendría a ser: ¿Cómo mejorar la precisión en el proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura mediante el uso de Aprendizaje Profundo?

1.1.2 Formulación de los Problemas Específicos

Se formulan dos problemas específicos que permiten abordar y resolver de manera detallada el problema general planteado en la presente investigación; estas son:

- Problema específico 1: ¿Cómo desarrollar un modelo de Aprendizaje Profundo que mejore la precisión del proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura?
- Problema específico 2: ¿Cómo validar la aplicación del modelo de Aprendizaje Profundo que es utilizado en la mejora de la precisión del proceso de detección automática de estados en superficies de pavimento flexible de la ciudad de Piura?

1.2 Objetivos

1.2.1 Formulación del Objetivo General

El objetivo general de la presente investigación se centra en abordar el problema general identificado de una manera específica y concreta en su solución. Por lo tanto, dicho objetivo se relaciona de manera directa con la problemática planteada en la investigación, siendo el siguiente: Determinar un modelo de Aprendizaje Profundo que permita mejorar la precisión en el proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura.

1.2.2 Formulación de los Objetivos Específicos

El objetivo y el problema general planteados se alcanzaron y resolvieron de manera sistemática y ordenada mediante los siguientes objetivos específicos:

- Objetivo Específico 1: Entrenar un modelo de Aprendizaje Profundo para mejorar la precisión del proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura.
- Objetivo Específico 2: Validar la aplicación del modelo de Aprendizaje Profundo para mejorar la precisión del proceso de detección automática de fallas en superficies de pavimento flexible en la ciudad de Piura en comparación con *Pavimenta2*.

1.3 Hipótesis y Variables

1.3.1 Formulación de la Hipótesis General

Se establece una relación entre las variables de acuerdo con su objetivo y el resultado que se espera lograr como hipótesis general de la investigación es: El uso del modelo de Aprendizaje Profundo mejora significativamente la precisión en el proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura, permitiendo detectar automáticamente los diferentes tipos de fallas con una precisión del 92.76 %.

1.3.2 Formulación de las Hipótesis Específicas

De la hipótesis general, formulamos las hipótesis específicas que dan respuesta a la presente investigación; estas son:

- Hipótesis Específica 1: El modelo de Aprendizaje Profundo mejora el proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura al detectar las fallas con una precisión igual al 92.76%.
- Hipótesis Específica 2: El modelo de aprendizaje profundo tendrá una métrica de evaluación F1-Score igual o superior al 96% en comparación con el software Pavimenta2, lo que valida la mejora de precisión del proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura.

1.3.3 Variables

Las variables predominantes de la investigación están relacionadas con el objetivo de utilizar un modelo de Aprendizaje Profundo para mejorar la precisión del proceso de detección automática de los estados de las superficies del pavimento flexible de la ciudad de Piura. Lo que permite evaluar el efecto directo del modelo de Aprendizaje Profundo en la precisión del proceso de detección automática; por tanto, se establecen como variable independiente y dependiente las que a continuación se mencionan:

- Variable independiente: Modelo de Aprendizaje Profundo.
- Variable dependiente: Precisión en el proceso de detección automática del estado de superficies de pavimento flexible.

1.4 Metodología

1.4.1 Tipos de Investigación

Según Hernandez et al. (2014), consideramos que la investigación tiene un enfoque cuantitativo, un diseño preexperimental, un nivel aplicativo y se fundamenta en un método de investigación hipotético-deductivo a partir de las siguientes razones.

1.4.1.1 Por Enfoque

La investigación se centra en el enfoque cuantitativo, se sustenta en que tiene variables definidas, utiliza datos numéricos (resolución y cantidad de imágenes) y busca analizar la relación entre las variables de manera objetiva y numérica, aunque el diseño sea preexperimental.

1.4.1.2 Diseño

Al establecer la variable independiente (el modelo de Aprendizaje Profundo) y la variable dependiente (la precisión en el proceso de detección automática del estado de las superficies de pavimento flexible), se están sentando las bases características de un diseño experimental.

Sin embargo, en un marco de control por parte de la investigación, se clasifica como preexperimental, al no contar con un grupo de control, lo que limita la validez interna. Asimismo, en atención a aquello, se realiza la comparación entre el modelo de Aprendizaje Profundo y el modelo de inteligencia artificial (software Pavimenta2), utilizando la misma cantidad de imágenes para el entrenamiento de ambos. Al no asignarse las imágenes de manera aleatoria a cada modelo, se concluye que su grado de control es mínimo. Para su análisis, el diseño se diagramó de la siguiente manera:

G ➔ X ➔ 0

G: Representa al grupo conformado para el entrenamiento del modelo.
X: Representa la evaluación de la variable dependiente mediante la ejecución de los modelos.
0: Representa los resultados de la evaluación de las imágenes en el tramo seleccionado.

1.4.1.3 Por nivel

La investigación se enfoca en la aplicación práctica del Aprendizaje Profundo para la detección automática de diferentes tipos de deterioros en las superficies de pavimento flexible. El objetivo principal es utilizar esta tecnología para mejorar la precisión del proceso de detección automática del estado de las superficies de pavimento flexible; por tanto, definimos que la investigación alcanza un nivel aplicativo.

1.4.1.4 Métodos

Según el método de investigación, el tipo de inferencia es hipotético-deductivo.

Hipotético: en el objetivo de la investigación, se plantea determinar un modelo aplicando algoritmos de Aprendizaje Profundo para mejorar la precisión en el proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura, sin conocer aún los resultados. Con base en ese objetivo, se plantea la hipótesis de que el modelo de Aprendizaje Profundo mejorará el proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura al detectar las fallas con una precisión igual al 92.76 %.

Deductivo: una vez que se obtengan los resultados de la evaluación del modelo de Aprendizaje Profundo, se procede a demostrar si la hipótesis planteada se cumple o no, es decir, si el modelo alcanza la precisión del 92.76 % en la detección de fallas, lo cual mejorará el proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura.

1.4.2 Población y Muestra

El muestreo no probabilístico o dirigido por conveniencia es adecuado para la investigación, ya que se han seleccionado intencionalmente las imágenes que cumplen con las características de los tipos de fallas definidos en el “Manual de carreteras: Mantenimiento o Conservación vial” del Ministerio de Transportes y Comunicaciones (MTC, 2018), del Perú, específicamente en el apartado de catálogo que propone los tipos de deterioros o fallas en la superficie de rodamiento de la calzada, es decir, la parte del pavimento por donde transitan los vehículos.

1.4.2.1 Población

La población de esta investigación serían todas las imágenes de las vías de pavimento flexible de la ciudad de Piura, extraídas de los videos grabados en las diferentes avenidas.

Todos los datos recopilados pasan por la etapa de “Preprocesamiento de los Datos Recopilados en Campo”, la cual se explica a detalle en el acápite 1.4.5. (Etapas metodológicas), a partir de ello nos enfocamos en las imágenes seleccionadas que cumplen con la homogeneidad por su tipo de falla, esto de acuerdo con las características e información detallada que menciona el “Manual de carreteras: Mantenimiento o conservación vial” (MTC, 2018). Las imágenes que fueron seleccionadas en el Pavimento Flexible de la ciudad de Piura fueron capturadas en un solo momento y pertenecen únicamente a los tres tipos de fallas superficiales que se evalúan en esta investigación.

Debido a limitaciones en cuanto a recursos y tiempo en la investigación, se realizó el muestreo a partir de esta población. Por lo tanto, se incluyeron los siguientes criterios de inclusión y exclusión:

- Criterios de inclusión: se seleccionan imágenes que presentan de manera clara los tres tipos de fallas establecidos en el catálogo "Deterioros o Fallas de los pavimentos flexibles", tales como fisura longitudinal, fisura transversal y piel de cocodrilo (MTC, 2018, p. 86).
- Criterios de exclusión: se excluyeron las imágenes que no mostraban claramente los tipos de fallas de interés especificados en el catálogo "Deterioros o Fallas de los pavimentos flexibles", así como aquellos que no concordaban con las características observadas en las “fotografías de referencia” (MTC, 2018, pp. 86-100).

1.4.2.2 Muestra

El tipo de muestreo utilizado es no probabilístico por conveniencia, ya que se han seleccionado las imágenes en función de su disponibilidad y adecuación a los criterios.

La unidad de muestreo y análisis son las imágenes de los tres tipos de fallas en el pavimento flexible.

La muestra final consta de 3312 imágenes, distribuidas de la siguiente manera:

- Fisura longitudinal: 1104 imágenes.
- Fisura transversal: 1104 imágenes.
- Piel de cocodrilo: 1104 imágenes.

La justificación del tamaño de muestra se fundamenta en que la cantidad de 3312 imágenes se estima como representativa de la población. Esto se debe a que se ha logrado recopilar un número considerable de imágenes para cada categoría de falla. Dicha cantidad de datos se considera suficiente para permitir un aprendizaje apropiado durante el entrenamiento del modelo. Por consiguiente, esto facilitará el logro de los objetivos propuestos en la investigación y posibilitará la obtención de conclusiones válidas.

1.4.3 Técnicas e instrumentos de recolección de datos

1.4.3.1 Técnicas de recolección de datos

Para la recolección de datos en este estudio, se optó principalmente por la técnica de observación en campo. Esta técnica resultó sumamente útil, ya que permitió obtener información de manera no intrusiva, sin interferir con el objeto de observación. En este caso, se utilizó grabaciones de video para registrar imágenes detalladas de la superficie del pavimento flexible en las vías de la ciudad de Piura. Esta técnica permitió analizar con detenimiento cada detalle y observar las imágenes en gabinete para su respectivo análisis.

Respecto a la selección de la red pre-entrenada *DenseNet-BC*, no se trata propiamente de un dato recolectado en sí mismo, sino de una elección realizada para utilizar una tecnología existente ya pre-entrenada en una gran cantidad de datos de imágenes para reconocer patrones y características de las imágenes, lo que permite ahorrar tiempo y recursos en el proceso de entrenamiento del modelo propuesto en esta investigación.

1.4.3.2 Instrumentos de recolección de datos

Los datos deben recopilarse con instrumentos que permitan adquirir y almacenar imágenes para entrenar el modelo de Aprendizaje Profundo y alcanzar los objetivos de la investigación.

Para recolectar los datos se usó sistemas de escaneo por área (fotografías) y video, capturados con un smartphone de alta gama (*Xiaomi 11T Pro*) desde un vehículo modelo *CHERY Q 22*.

Para registrar la cantidad de imágenes seleccionadas, se empleó un formato específico detallado en el Anexo 4, el cual posibilitó la clasificación de las imágenes según el tipo de falla observado en el pavimento flexible. Con esto, se logró contabilizar y organizar de manera precisa todas las imágenes seleccionadas, lo que simplificó su almacenamiento en una base de datos estructurada por el tipo de falla correspondiente.

Además, es importante mencionar que la red pre-entrenada se utilizó en conjunto con la técnica de aprendizaje transferido para adaptarla a las características específicas de los tipos de fallas en el pavimento flexible de la ciudad de Piura.

1.4.4 Técnicas e Instrumentos de Análisis y Procesamiento De Datos

1.4.4.1 Técnicas de análisis y procesamiento de datos

Preprocesamiento de imágenes para ajustar y mejorar la calidad de las imágenes adquiridas en campo, como por ejemplo eliminar ruido.

Modelado de Aprendizaje Profundo (procesamiento de imágenes).

Análisis estadístico del entrenamiento del modelo de Aprendizaje Profundo, evaluación del modelo mediante el cálculo de probabilidades y la validación mediante el porcentaje de coincidencias.

1.4.4.2 Instrumentos de análisis y procesamiento de datos

Para extraer todas las imágenes se automatizó con scripts ejecutados en *Python*, y para los recortes de limpieza de ruido algunos se realizaron en Python, pero la gran mayoría se hizo en el software de edición de imágenes *Adobe Photoshop* versión del 2021 mediante procesos semiautomáticos. Así también, este último programa se utilizó para el aumento del conjunto de datos.

Se empleó un ambiente de desarrollo para programar en el lenguaje Python. Se utilizó la biblioteca *TensorFlow*, la cual es un *framework* de código abierto completo que permite realizar tareas importantes, tales como el modelado y entrenamiento de redes neuronales. Para realizar tareas específicas de procesamiento de imágenes y videos, como la detección y clasificación de objetos, se empleó *ImageAI*, una biblioteca que complementa a TensorFlow para el modelado en Aprendizaje Profundo.

La métrica de precisión de prueba (*test accuracy*) se monitorea a lo largo de las épocas de entrenamiento. El valor máximo de esta métrica se registra y se considera la mejor precisión de prueba (*best test accuracy*). En esta investigación se presentan mediante gráficos lineales.

La evaluación del modelo se realiza con imágenes nuevas, es decir, imágenes que no formaron parte del conjunto de entrenamiento ni del conjunto de pruebas. Cuando se clasifica una imagen nueva utilizando el modelo, este calcula la probabilidad que el algoritmo le da a cada una de las categorías de clasificación y finalmente asigna la categoría con la mayor probabilidad.

Se valida el modelo de Aprendizaje Profundo al comparar la precisión de detección automática de fallas con el software Pavimenta2 mediante el porcentaje de coincidencias en tablas.

1.4.5 Etapas metodológicas

Con el objetivo de garantizar un enfoque completo y preciso, se ha decidido seguir una metodología en etapas, teniendo en cuenta las siguientes premisas. Durante las tres primeras etapas se recolectarán imágenes precisas y relevantes, se les aplicará un proceso de preprocesamiento y se diseñará la estructura de la red neuronal que finalmente se ejecutará en el entrenamiento en la cuarta etapa. A continuación, se detallan las etapas con sus respectivos pasos:

Etapa 1. Recopilación de Datos de Campo:

- Se grabaron vídeos con la cámara principal de 108 megapíxeles del smartphone, colocada en el parabrisas delantero del vehículo.
- Se recorrieron tramos de las avenidas Miguel Grau, Don Bosco (ex Circunvalación), Andrés Avelino Cáceres, Cesar Vallejo y Sánchez Cerro de la ciudad de Piura. Exactamente, como se aprecia en el Anexo 5.

- Las velocidades en que se recorrieron las avenidas seleccionadas oscilaron entre los 25 y los 35 kilómetros por hora. Además, se procuró mantener una velocidad constante a lo largo de cada recorrido de las vías evaluadas. De este modo, se pudo garantizar la uniformidad en la recolección de los datos necesarios para el análisis correspondiente.
- Todas las grabaciones de video han sido configuradas para que alcancen una velocidad de fotogramas máxima de 30 FPS, lo que significa que obtienen hasta 30 fotografías por cada segundo de grabación.
- Se procede a extraer todas las imágenes capturadas tanto en la ida como en la vuelta del tramo recorrido para su posterior evaluación.

Etapa 2. Preprocesamiento de los Datos Recopilados en Campo:

- Se extrajeron las imágenes del video de cada avenida de acuerdo con su recorrido. Posteriormente, se seleccionaron aquellas imágenes que presentaban las tres fallas de interés. Estas imágenes tienen una resolución de 1080 x 1920 píxeles, tal como se especifica en el Anexo 4.
- En el proceso de selección, se observó que muchas de las imágenes presentaban baja calidad debido al ruido presente. Como medida, se procedió a recortar todas las imágenes seleccionadas a un tamaño de 800 x 450 píxeles. Esta acción permitió la eliminación del ruido y la aplicación de un segundo filtro de mayor calidad, garantizando así que solo se incluyeran imágenes que cumplían con los criterios de inclusión establecidos.
- Se identificó que las fotografías presentaban un alto nivel de ruido, lo cual podría impactar negativamente en el proceso de entrenamiento. Además, las imágenes eran demasiado pesadas. Por consiguiente, se realizó un recorte selectivo, centrándose únicamente en la extracción de la información relevante sobre la falla. Esta acción posibilitó mejorar tanto la calidad como el tamaño de las imágenes, lo que resultó en una optimización del proceso de aprendizaje.
- Después de completar el proceso de reducción de ruido en los recortes, se procedió a estandarizar el tamaño de las imágenes alrededor de 200 píxeles de ancho y alto. De esta manera, se aseguró de que todas las imágenes tuvieran la misma resolución y calidad aproximada. Esta decisión se tomó para garantizar una alta precisión y aprovechar al máximo la capacidad informática de las *GPU* y el algoritmo pre-entrenado disponible para esta investigación.

- Al contar con una cierta cantidad de imágenes que cumplían con los criterios de inclusión y reducir el ruido significativamente, se procedió al aumento del conjunto de datos, que también es conocido como *data augmentation*, que es el proceso de aumentar la cantidad de imágenes disponibles para el entrenamiento.
- Al final, se guardaron las imágenes de acuerdo con su tipo de falla y se clasificaron para el entrenamiento en carpetas de nombre *train* y *test*. Al tener la base de datos definida, se procede a diseñar la estructura de la red neuronal base.

Etapa 3. Diseño de Estructura de Red Neuronal Base:

- En síntesis, se elaboró el script en el entorno de Python con la tarea principal de entrenar el nuevo modelo de Aprendizaje Profundo con la arquitectura de red neuronal convolucional seleccionado como algoritmo base para aplicar aprendizaje transferido con el conjunto de datos de imágenes de fallas del pavimento flexible capturadas por la cámara del smartphone de esta investigación.

La siguiente etapa se enfoca en generar la respuesta de la primera pregunta específica y con ello validar su hipótesis.

Etapa 4. Entrenamiento de Modelo Propuesto:

- Se dividió el conjunto de entrenamiento en dos partes: train y test, como se mencionó previamente.
- Antes de entrenar el modelo, se debe asegurar de tener una tarjeta gráfica compatible. Se importa la red pre-entrenada y se entrena con los datos de entrenamiento. Se ajustan los hiper-parámetros, como el número de épocas y el tamaño de los lotes (*Batch Size*) para lograr la precisión requerida.
- Resulta crucial someter el modelo de Aprendizaje Profundo a una evaluación exhaustiva a través de un conjunto de pruebas rigurosas. El objetivo es lograr predicciones precisas en nuevas fotografías. Este procedimiento se llevará a cabo una vez que los resultados del entrenamiento hayan convergido en torno a la precisión de la hipótesis planteada.
- Posteriormente, se incluyeron códigos para detectar y delimitar las fallas de cada tipo en estudio. Asimismo, se llevaron a cabo las pruebas.
- Finalmente, se procede a realizar la prueba en un tramo seleccionado y se comparan los resultados con otro modelo de inteligencia artificial del software de Pavimenta2, utilizando la misma base de datos de entrenamiento.

Después de finalizar la implementación del modelo propuesto por esta investigación en el análisis, surgió la pregunta: ¿Cómo validar el software con el modelo de Aprendizaje Profundo propuesto?

Etapa 5. Realización de Pruebas y Comparación de Resultados:

- Para la selección del conjunto de validación, se eligió un tramo específico de aproximadamente 1 km ubicado sobre la superficie de pavimento flexible a lo largo de la Avenida Don Bosco, el cual presenta fallas. La información se grabó en vídeo y, posteriormente, se extraerán imágenes con el objetivo de realizar la evaluación.
- La evaluación de las mismas fotografías a procesar no tiene ningún filtro, solo se proceden a evaluar, ya que la detección será automática por cada modelo. De esta forma, se asegura la objetividad y transparencia en la evaluación de los resultados.
- En última instancia, se procede a comparar los resultados del software desarrollado con el nuevo modelo de Aprendizaje Profundo con los resultados previos de Pavimenta2 para evaluar la precisión del enfoque propuesto y su potencial aplicación práctica.

Etapa 6. Redacción y Sustentación de la Tesis de Maestría:

- En esta última etapa del proceso, se analizarán los resultados obtenidos al comparar las métricas de evaluación entre el software desarrollado y Pavimenta2. Este análisis posibilitará determinar si el software categoriza las fallas de manera similar a Pavimenta2, validando así la precisión en la detección automática de los estados de las superficies de pavimento flexible. Una vez logrados los objetivos propuestos, se procederá a la redacción final de la tesis. Posteriormente, se llevará a cabo la presentación, se atenderán las observaciones planteadas y se realizará la sustentación final de la tesis.

CAPÍTULO II: MARCO TEÓRICO

2.1 Antecedentes de la Investigación

Se realizó una búsqueda exhaustiva y revisión minuciosa de textos y artículos científicos publicados en revistas indexadas, que abordan los objetivos de este trabajo de investigación.

Esta búsqueda condujo a encontrar numerosas investigaciones relacionadas con la detección y clasificación automatizadas de fallas en pavimentos, las cuales han arrojado resultados significativos. En consecuencia, este estudio se enfoca en cuatro etapas fundamentales: la adquisición de datos, el preprocesamiento y procesamiento de imágenes mediante el método de Aprendizaje Profundo, y como quinta etapa, la validación de la métrica de evaluación a través de la realización de pruebas y la comparación de resultados.

Por lo tanto, se presentan las investigaciones relevantes que respaldan las cuatro etapas mencionadas anteriormente, al tiempo que se destaca que la quinta etapa constituye una incorporación novedosa en esta investigación.

2.1.1 Antecedentes a nivel Internacional

- *"How to Get Pavement Distress Detection Ready for Deep Learning? A Systematic Approach"*, Eisenbach et al. (2017): en este estudio, se preparó una base de datos libre que permite desarrollar una extensa evaluación de redes neuronales profundas modernas, logrando evaluar el estado de arte en la detección de fallas en el pavimento de manera significativa mediante un proceso estandarizado que cumple con las regulaciones federales Alemanas. Analizando la eficacia de técnicas tales como el *dropout*, *batch normalization* , *max-norm regularization* y *weight decay*. Los mejores resultados de los algoritmos para evaluar la regularización se lograron utilizando los métodos de dropout, seguido de batch normalization y el que disminuyó notablemente el rendimiento fue *large weights*.

- *"Road damage detection and classification using deep neural networks with smartphone images",* Maeda et al. (2018): esta investigación aporta contribuciones relevantes en relación a los temas planteados. Primero, se prepara a gran escala un conjunto de datos de 9053 imágenes de daños en la carretera capturadas con un teléfono inteligente instalado en un vehículo, abarca 15,435 casos de fallas en la superficie de la carretera. En segundo lugar, el conjunto de datos se entrena con

un método de detección de daños innovador basado en redes neuronales convolucionales, de ello se compara la precisión en el tiempo de ejecución entre un servidor con GPU y un teléfono inteligente. Por último, se demuestra que el tipo de daño se logra clasificar en ocho categorías con alta precisión empleando el método propuesto.

- *"Feature pyramid and hierarchical boosting network for pavement crack detection"*, F. Yang et al. (2019): se propone una arquitectura de red novedosa, denominada *Feature Pyramid* and *Hierarchical Boosting Network* (*FPHBN*) que se basa en técnicas de Aprendizaje Profundo y también se sugiere una medida novedosa para la detección de grietas llamada *Average Intersection Over Union* (*AIU*). Se compara FPHBN a otros métodos de detección de grietas basados en técnicas de Aprendizaje Profundo como *Holistically-Nested Edge Detection* (*HED*), *Richer Convolutional Features* (*RCF*), *Fully Convolutional Networks* (*FCN*) y este último *CrackForest* que está basado en métodos tradicionales que se entiende en esta investigación como técnicas de aprendizaje no profundo. En resumen, los métodos mencionados son utilizados con cinco conjuntos de datos de grietas (imágenes) que son conocidos como *CRACK500*, *GAPs384*, *Cracktree200*, *CFD* y *AEL*; finalmente como resultado se obtuvo que el valor de AIU del método propuesto FPHBN tiene un mejor rendimiento en comparación que HED, RCF, FCN y CrackForest.

- *"A method of data augmentation for classifying road damage considering influence on classification accuracy"*, Tsuchiya et al. (2019): esta investigación propuso un método para el aumento de datos de aprendizaje del conjunto de datos de daños en la carretera teniendo en cuenta la influencia de los datos aumentados en la precisión de la detección y clasificación. Los métodos efectivos de aumento de datos dependen de la clase. Se seleccionan en un inicio dieciséis conjuntos de datos y cinco métodos de aumento de datos para cada clase de forma independiente. El conjunto de datos de aprendizaje óptimo se genera mediante la aplicación de métodos efectivos de aumento de datos. *YOLOv3* es una inteligencia artificial que ha sido altamente entrenada y capacitada y que funciona con el conjunto de datos generados y obtiene muy buenos resultados. La evaluación cuantitativa realizada por los autores mostró que YOLOv3 entrenado con el conjunto de datos generado por el método propuesto obtiene un error de precisión media (mAP) de 27.41% más alto que el entrenado con el conjunto de datos original.

- *"Detection and classification of road damage using R-CNN and faster R-CNN: a deep learning approach"*, Arman et al. (2020): el estudio investigó sobre la detección y clasificación de daños en la carretera a partir de imágenes de superficies de la carretera utilizando el método de detección de objetos. Aplicó el algoritmo de red neuronal convolucional múltiple (*CNN*) para clasificar los daños en las carreteras e identificó qué algoritmo funciona mejor en la detección y clasificación de esa tarea. Los daños se clasifican en tres categorías: bache, fisura y destape. La recopilación de información se llevó a cabo a través de capturas fotográficas de las calles de Dhaka utilizando la cámara de un dispositivo móvil, para luego preparar los datos mediante ajustes como el redimensionamiento de las imágenes, el balance de blancos, la modificación de contrastes y la adecuada asignación de etiquetas. En esta investigación, se implementaron los modelos de detección de objetos *R-CNN* y *FASTER R-CNN* para identificar los daños en las carreteras, mientras que para la clasificación se empleó *Support Vector Machine* (SVM). Los resultados demuestran que la precisión sobresaliente del 98.88% se logró con FASTER R-CNN.

- *"Deep learning-based road damage detection and classification for multiple countries"*, Arya et al. (2021): en esta investigación, los autores exploraron con un conjunto de datos de daños viales heterogéneos a gran escala que comprende 26,620 imágenes recopiladas de carreteras de varios países. Concluyeron que el rendimiento de los modelos de Aprendizaje Profundo para detectar y clasificar estos daños se ve afectado por el país de origen de las imágenes. No obstante, demostraron que es factible desarrollar un modelo eficiente de detección de daños en las carreteras de un país mezclando los datos locales de algún otro país. En esencia, este trabajo sienta las bases para crear un modelo estandarizado de detección y clasificación de daños en carreteras que se aplique a nivel mundial, lo cual sería útil para ingenieros, autoridades viales y municipios de diferentes países.

- *"An efficient method for detecting asphalt pavement cracks and sealed cracks based on a deep data-driven model"*, N. Yang y Li (2022): en este estudio, los investigadores presentaron cuatro contribuciones. En primer lugar, propusieron un nuevo método de anotación denso y redundante para detectar las características estructurales de las grietas en el asfalto y las grietas selladas. En segundo lugar, desarrollaron un conjunto de datos que contenía 10,400 imágenes de pavimento basadas en este nuevo enfoque de anotación. En tercer lugar, utilizaron un método semiautomático de etiquetado de grietas y grietas selladas, lo que ahorró un 80%

del tiempo en comparación con el etiquetado manual, mejorando significativamente la eficiencia del proceso de detección de grietas. Finalmente, evaluaron 13 modelos de detección de objetos populares, y los resultados demostraron que los métodos de etiquetado denso y redundante eran efectivos. En resumen, los modelos de la serie *YOLOv5* mostraron un mejor rendimiento y un equilibrio más adecuado en comparación con los demás, siendo *YOLOv5s* el mejor con una puntuación F1 del 86.79% y un tiempo de inferencia de 14.8 ms. Esta tecnología nos permite identificar estos defectos con precisión, lo que resulta crucial para el mantenimiento y la reparación adecuada de las carreteras.

2.1.2 Antecedentes a nivel Nacional

En el Perú, no se cuenta con información precisa sobre el inicio del uso de la tecnología prometedora de Inteligencia Artificial, especialmente en el subcampo de Aprendizaje Profundo. A pesar de ello, se tiene conocimiento de que su implementación ha ido en aumento en los últimos años. Se han identificado estudios independientes relacionados con pavimentos flexibles realizados en Universidades Peruanas, aunque estos estudios son aún escasos. Algunos de los destacados son:

- "Influencia de la rigidez del pavimento en la condición superficial del pavimento utilizando técnicas de inteligencia artificial en la vía nacional PE-26B", Esplana y Pérez (2021): en la presente investigación, se planteó el objetivo de analizar cómo la rigidez del pavimento afecta la condición superficial de la vía nacional PE-26B, tramo Huancavelica - Lircay. Se identificaron 70 unidades muestrales para recopilar datos del número estructural corregido (*SNC*) y el índice de condición del pavimento (*PCI*). A través de estadísticas descriptivas, se evaluaron los datos para determinar la correlación entre las variables. Posteriormente, se utilizó redes neuronales artificiales para analizar esta dependencia. El resultado fue un modelo inteligente que predice el valor del índice de condición del pavimento (PCI) utilizando el dato de entrada del número estructural corregido (SNC). El modelo se ajustó bien a los datos y demostró una correlación positiva significativa con un coeficiente de determinación de R2 = 0.9127. Así, se concluyó que el número estructural corregido influye de manera considerable en la condición del pavimento. En resumen, la rigidez del pavimento tiene un impacto sustancial en la condición superficial del mismo, demostrando la eficacia del uso de técnicas de inteligencia artificial en la vía nacional PE-26B.

- "Deep learning para la detección de fallas en pavimentos de una zona del distrito de Villa María del Triunfo 2022", Zúñiga (2022): la tesis de investigación tiene como objetivo desarrollar un sistema de inteligencia artificial basado en Aprendizaje Profundo para detectar fallas en pavimentos de asfalto y concreto en una zona de Villa María del Triunfo. Para ello, se utilizaron las metodologías de *CommonKAD'S* y el algoritmo YOLOv5, junto con Python y diversas bibliotecas. Se trabajó con una muestra de 420 imágenes que mostraban fallas como grietas y huecos, capturadas con un teléfono inteligente, de las cuales se analizaron 201. En la fase de pruebas (test), se analizaron 30 imágenes con un nivel de confianza promedio del 0.62 % después de 724 iteraciones. Los resultados muestran una precisión del 0.93 % en la detección de grietas y del 0.77 % en la detección de huecos, con una sensibilidad del 0.92 % y 0.91 % respectivamente. La puntuación final fue del 0.86 % para grietas y del 0.77 % para huecos.

2.1.3 Antecedentes a nivel Local

No se han encontrado estudios en la localidad que estén relacionados con la temática de la investigación.

2.2 Bases Teóricas

En esta sección, se hace referencia al concepto de pavimento flexible y sus tipos de fallas, enfocándose solo en aquellas que son objetivo de detección en esta investigación. Además, se explica la teoría del método de Aprendizaje Profundo y sus principales enfoques relacionados con esta tesis.

2.2.1 Pavimento Flexible

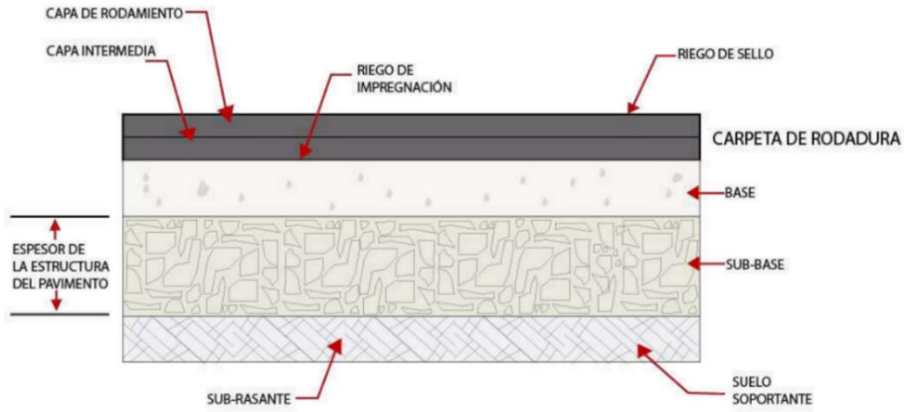
De acuerdo con el "Manual de Carreteras: Suelos, Geología, Geotecnia y Pavimentos" del Ministerio de Transportes y Comunicaciones (MTC), el pavimento flexible consiste en una estructura multicapa que incluye una capa de rodadura asfáltica sobre capas granulares de base y sub-base.

Esta estructura debe proporcionar una superficie uniforme y resistente al tráfico, condiciones climáticas y otros factores, distribuyendo adecuadamente las cargas a las capas inferiores (MTC, 2014).

En la Figura 1 se muestra cómo está compuesta por lo general la estructura de un pavimento flexible, y estas se detallan a continuación:

Figura 1

Estructura del pavimento flexible



Nota: Adaptado de (MTC, 2014).

- Subrasante: es la capa más extensa y profunda que sirve de base para la estructura del pavimento. Inicialmente, corresponde a la superficie finalizada de la carretera tras las operaciones de movimiento de tierras, ya sea por corte o relleno.
- Sub-base: es una capa de material diseñada para drenar el agua y dar soporte a la base y la superficie de rodadura.
- Base: en la estructura, la capa inferior cumple la función de sostener, distribuir y transmitir las cargas sometidas por el tránsito. Se trata con asfalto, cemento o cal.
- Carpeta de rodadura: es la capa superficial del pavimento flexible, situada sobre la base para asegurar una buena adherencia. Esta capa es la que directamente soporta las cargas generadas por el tránsito vehicular.

2.2.2 Clasificación de los diferentes tipos de fallas de los pavimentos flexibles

El “Manual de Carreteras: Mantenimiento o Conservación Vial” del MTC (2018), clasifica los deterioros o fallas del pavimento flexible en dos categorías, denominadas “estructurales” y “superficiales” (p. 86).

En el caso de la investigación, se centra exclusivamente en la tipología de la falla, ya que la clasificación en las categorías abordaría un tema aparte de discusión y el enfoque principal es la detección de acuerdo con el tipo de falla en el pavimento flexible; por tanto, de acuerdo con la tipología de falla propuesta en el “Manual de Carreteras: Mantenimiento o Conservación Vial” del MTC (2018), se identifican en un pavimento flexible las fallas mencionadas a continuación:

- Piel de cocodrilo.

- Baches (Huecos).
- Deformación por deficiencia estructural.
- Ahuellamiento.
- Reparaciones o Parchados.
- Peladura y Desprendimiento.
- Fisuras longitudinales.
- Fisuras transversales. (p. 86)

2.2.2.1 Tipología de fallas y su detección

De acuerdo con el "Manual de Carreteras: Mantenimiento o Conservación Vial" (MTC, 2018), se han seleccionado las siguientes fallas de pavimento flexible de interés para la investigación:

- Falla piel de Cocodrilo: este deterioro o falla se manifiesta en la capa asfáltica del pavimento flexible con la característica particular de que las fisuras y/o grietas son ramificadas en forma de escamas irregulares que se asemejan a la piel de un cocodrilo, como se observa en la figura 2. Estas grietas son superficiales, pero también indican problemas subyacentes en la estructura del pavimento, como la fatiga del material debido al tráfico y a las condiciones climáticas. La presencia de la falla "piel de cocodrilo" compromete la durabilidad y capacidad de carga del pavimento flexible, por lo que su detección y reparación oportuna son fundamentales para mantener la integridad en las vías a nivel nacional (MTC, 2018).

Figura 2

Fisura piel de cocodrilo

- Fisuras longitudinales: este deterioro o falla se manifiesta en la capa asfáltica del pavimento flexible como fisuras y/o grietas que se extienden en la dirección del flujo de tráfico, como se observa en la Figura 3. En la mayoría de los casos, estas grietas

son causadas por fatiga del material y problemas en la base del pavimento que indican insuficiencia estructural del pavimento. Por consiguiente, detectar y reparar este tipo de falla es fundamental para mantener la infraestructura vial en condiciones óptimas (MTC, 2018).

Figura 3

Fisura longitudinal

- Fisuras transversales: son fracturas transversales del pavimento que se presentan de manera perpendicular (o cercana) al eje de la vía, como se observa en la Figura 4. Este tipo de falla generalmente proviene de la retracción térmica de la mezcla asfáltica, la cual pierde flexibilidad debido a un exceso de material de relleno o al envejecimiento del asfalto. También puede deberse a la propagación de grietas desde capas inferiores y a la apertura de juntas de construcción que no fueron adecuadamente selladas (MTC, 2018).

Figura 4

Fisuras transversales

2.2.3 Aprendizaje Profundo

2.2.3.1 Acontecimientos y Cronología

La historia del Aprendizaje Profundo se encuentra en un campo anteriormente conocido como cibernética. Este campo comenzó a desarrollarse en 1943 con la publicación de un artículo de *McCulloch* y *Pitts*, en el que trataron de explicar el funcionamiento del sistema nervioso.

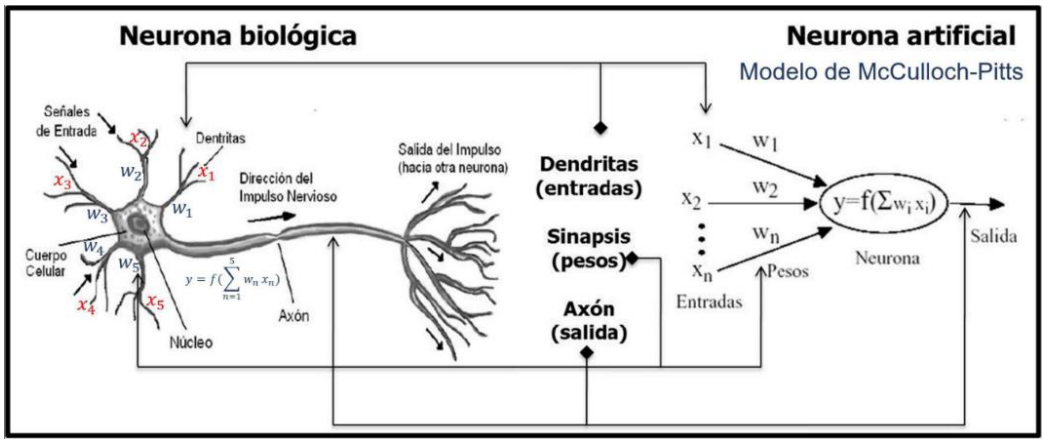
En la Figura 5, se observa la representación matemática del modelo de neurona artificial de McCulloch-Pitts. De esta comparación, se concluye que el objetivo de una neurona artificial es imitar el comportamiento de una neurona biológica. La interacción entre estas neuronas artificiales da lugar a la formación de redes neuronales artificiales.

La Figura 6 presenta una cronología que proporciona una representación visual de los eventos que han consolidado los estudios en el campo de las redes neuronales artificiales a lo largo del tiempo. Este proceso ha llevado al desarrollo de lo que hoy se conoce como Aprendizaje Profundo o Deep Learning, en inglés.

Finalmente, desde hace unos años Hinton et al. (2006) demostraron la creación de una nueva red neuronal llamada Red de creencia profunda, en su artículo titulado *"A fast learning algorithm for deep belief nets"*. Este tipo de red neuronal hizo que el proceso de entrenamiento con grandes cantidades de datos fuera más sencillo. A partir de ese año, se ha venido utilizando el término Deep Learning, el cual se basa en una estructura de red neuronal que pretende emular el aprendizaje biológico.

Figura 5

Comparación entre una neurona biológica y una neurona artificial



Nota: Adaptado de (Lao et al., 2017).

Figura 6

Cronología de momentos importantes en la historia del aprendizaje profundo

Nota: Adaptado de (Hinton et al., 2006).

2.2.3.2 Definiciones generales.

Según Chatterjee (2020), el concepto de inteligencia artificial se refiere al término fundamental empleado para describir la capacidad que posee una computadora o máquina para ejecutar y gestionar determinadas actividades o decisiones.

Por otro lado, Saleh (2019) se enfoca en las características distintivas de la inteligencia artificial, las cuales son: la capacidad de predicción y adaptación (con el uso de algoritmos que identifican patrones a partir de grandes cantidades de información), la toma de decisiones de forma autónoma (la habilidad de aumentar la inteligencia humana, brindar información y mejorar la productividad), el aprendizaje continuo y la predicción del futuro.

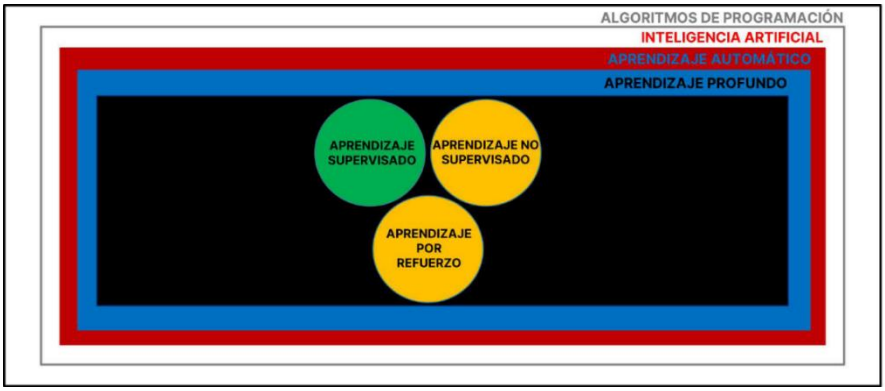
Una rama importante de la inteligencia artificial es el aprendizaje automático que se centra en el uso de datos y algoritmos para modelar cómo aprenden los humanos, aumentando gradualmente su precisión (Chatterjee, 2020).

Por tanto, el aprendizaje automático se refiere a métodos que detectan automáticamente patrones en bases de datos, y utilizan estos patrones para predecir información futura o tomar decisiones bajo incertidumbre (Murphy, 2012).

Finalmente, es necesario definir el término de Aprendizaje Profundo, que es un subcampo del aprendizaje automático dentro del campo de la inteligencia artificial, como se muestra en la Figura 7. Los algoritmos de Aprendizaje Profundo se organizan de manera estructurada o jerárquica, emulando el proceso de aprendizaje humano con el objetivo de adquirir conocimientos específicos. Constituye una función de aprendizaje automático que opera mediante un proceso de toma de decisiones no lineal. Esto permite a los modelos computacionales, compuestos por múltiples capas de procesamiento, aprender representaciones de datos con diversos niveles de abstracción (Chatterjee, 2020).

Figura 7

Organización y relación de los subconjuntos con la inteligencia artificial



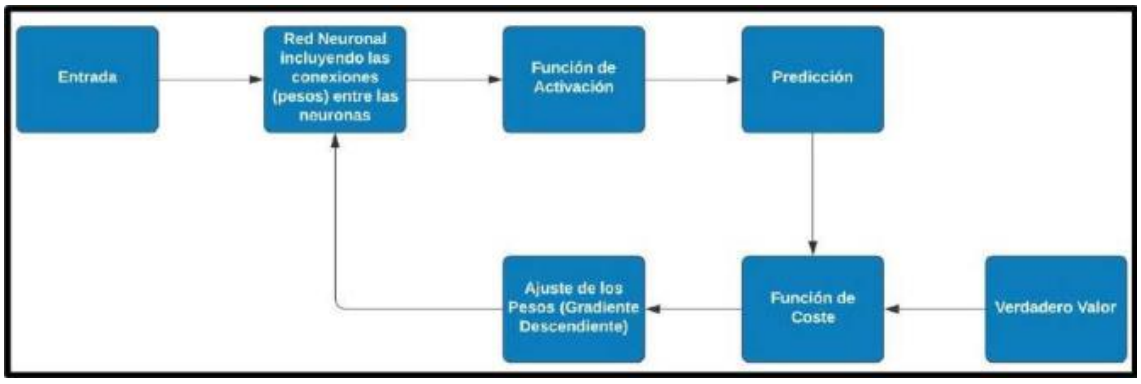
2.2.3.3 Principios y elementos que lo componen.

El Aprendizaje Profundo utiliza algoritmos más complejos que el Aprendizaje Automático, centrándose en el uso de redes neuronales artificiales para imitar el funcionamiento de las redes neuronales biológicas. Sus principales aplicaciones son tareas de clasificación, como reconocimiento de voz y clasificación de imágenes (Manzanares, 2019).

En la Figura 8 se muestra el diagrama de flujo del proceso simplificado de Aprendizaje Profundo de una red neuronal. En el centro de cada modelo de Aprendizaje Profundo, hay una red neuronal artificial. Principalmente, una red neuronal artificial está compuesta por un número variable de capas de neuronas, y la información se desarrolla viajando de capa en capa hasta llegar a la capa final (Montalvo, 2021).

Figura 8

Diagrama de flujo del proceso de aprendizaje profundo de una red neuronal



Nota: Adaptado de (Montalvo, 2021).

Ahora presentamos a detalle cada elemento que compone y participa en el proceso de Aprendizaje Profundo:

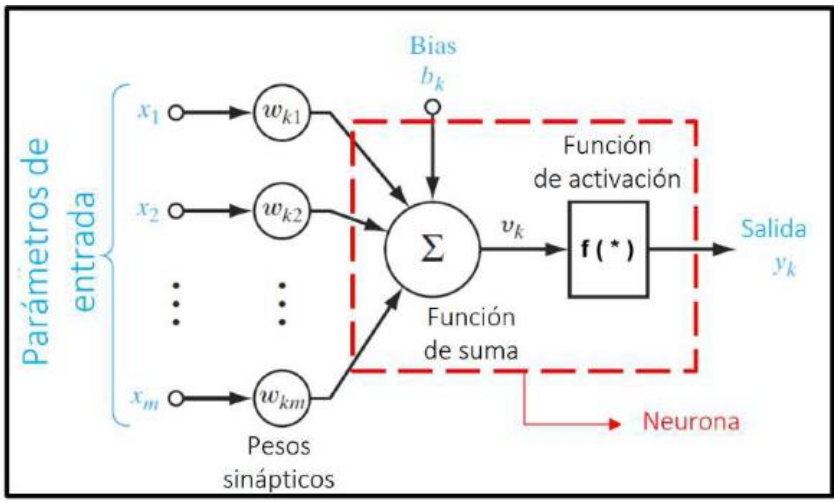
2.2.3.3.1 Redes Neuronales artificiales.

Las redes neuronales artificiales, o traducidas al inglés *artificial neural networks*, son conocidas por sus siglas (*ANN*), y consisten en modelos computacionales capaces de procesar aprendizajes mediante unidades interconectadas que sirven como neuronas artificiales. Esto está inspirado en el proceso de aprendizaje del cerebro humano (Montalvo, 2021).

En la Figura 9 se muestra una red neuronal artificial más básica de las ANN conocida por perceptrón simple, y su estructura se compone de tres elementos principales: Primero recibe unas entradas (x) que suelen ser del exterior o de otras neuronas de la red que estuvieran interconectadas. Segundo, un sumador que suma las señales de entrada ponderadas mediante sinapsis o enlaces de conexión.

Figura 9

Estructura típica de una neurona artificial



Nota: Adaptado de (Haykin, 1998).

Cada entrada está asociada a un peso (W), y mientras mayor sea este peso, mayor importancia tendrá la entrada asociada a este peso. Tercero y último, tenemos una función de activación que limita el resultado de la neurona transformando las entradas y computa la salida (Y).

La estructura matemática de una neurona se demuestra mediante la Ecuación 1:

$$v_k = \sum_{j=i}^{m} w_{kj} x_j \qquad (1)$$

La Ecuación 2 calcula la salida Y_k, luego de haber realizado la sumatoria, al resultado final obtenido se le aplica una función de activación $f(*)$.

$$Y_k = f(v_k + b_k) \qquad (2)$$

Donde: x_j son señales de entrada.

w_{kj} son pesos sinápticos.

v_k es la salida del sumador.

b_k es el sesgo o bias.

$f(*)$ es la función de activación.

Y_k es la señal de salida de la neurona.

2.2.3.3.2 Función de activación

Las funciones de activación son esenciales porque determinan el comportamiento de cada neurona en una red neuronal. Introducen no linealidad al procesar las entradas, permitiendo que la red aprenda y realice tareas más complejas.

Una red neuronal sin una función de activación es esencialmente solo un modelo de regresión lineal. Aunque existen una gran variedad de funciones, a continuación, se presentan algunas de las funciones de activación típicas no lineales, mencionadas por Montalvo (2021).

En resumen, en la Tabla 1, se detalla el tipo de función, su fórmula, las ventajas y desventajas de cada una de ellas.

En la Figura 10 se muestra el diagrama de la función sigmoidea, la Figura 11 representa la función tangente hiperbólica, la Figura 12 representa la función de la *Unidad Lineal Rectificada*, también conocida como *ReLU* y la Figura 13 representa la función *Leaky ReLU* con $\alpha = 0.1$, todas ellas correspondientes a las funciones de la Tabla 1.

Tabla 1

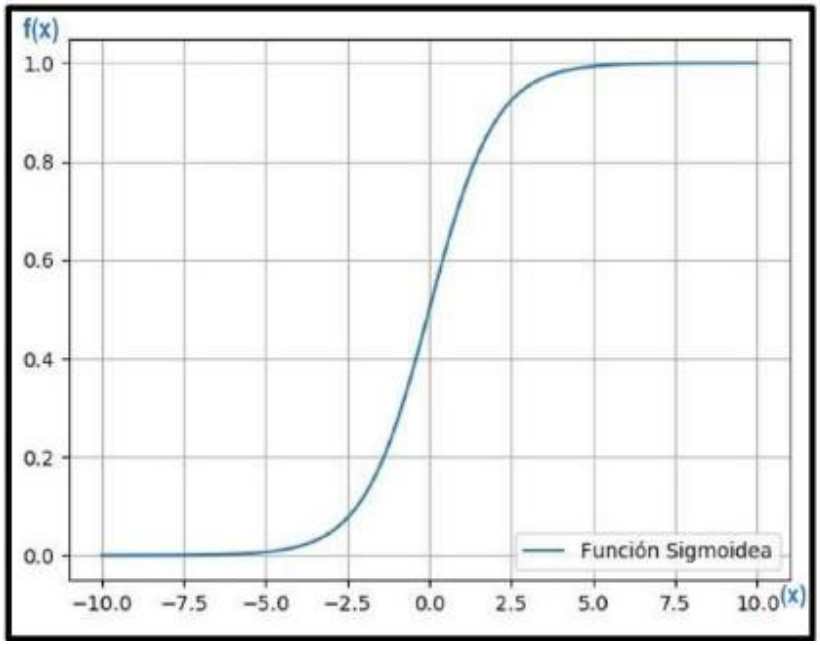
Ventajas y desventajas de las funciones de activación

Función de activación	Fórmulas	Ventajas	Desventajas
Sigmoidea	$f(x) = \frac{1}{1+e^{-x}}$ (3) Donde: $f(x)$ es la función sigmoidea. e es la constante de Euler. x es la entrada de la función.	▪ La función sigmoide se utiliza por defecto en las neuronas de la capa de salida de los modelos de clasificación binaria, ya que su salida se interpreta como probabilidades. ▪ Fácil de entrenar sobre un conjunto de datos pequeños.	▪ Problemas de saturación del gradiente. ▪ La salida no está centrada en 0 y ocasiona el covariate shift.
Tangente hiperbólica	$f(x) = \tan h\,(x)$ (4) Donde: $f(x)$ es la función tangente hiperbólica. x es la entrada de la función.	▪ La salida está centrada en 0.	▪ Problemas de saturación del gradiente. ▪ Difícil de entrenar en conjuntos de datos pequeños.
ReLU	$f(x) = \begin{cases} x\ if\ x \geq 0 \\ 0\ if\ x < 0 \end{cases}$ (5) Donde: $f(x)$ es la función ReLU. x es la entrada de la función.	▪ Aprende mucho más rápido que la función sigmoidea y Tangente hiperbólica. ▪ Evita y rectifica los problemas de saturación del gradiente.	▪ Solo se utiliza en las capas ocultas. ▪ Es difícil de entrenar en conjuntos de datos pequeños. ▪ Necesita de muchos datos para aprender un comportamiento no lineal. ▪ La mitad del rango el gradiente es 0. Ocasionando que el sesgo se desplace lentamente.
Leaky ReLU	$f(x) = \begin{cases} x\ if\ x \geq 0 \\ \alpha x\ if\ x < 0 \end{cases}$ (6) Donde: $f(x)$ es la función Leaky ReLU. x es la entrada de la función. α es un factor que se utiliza para permitir los valores negativos.	▪ Resuelve el problema de la ReLU del medio gradiente 0.	▪ Necesita ajustar los parámetros para encontrar un buen parámetro de caída lenta. ▪ Comúnmente no se utiliza.

Nota: Adaptado de (Montalvo, 2021).

Figura 10

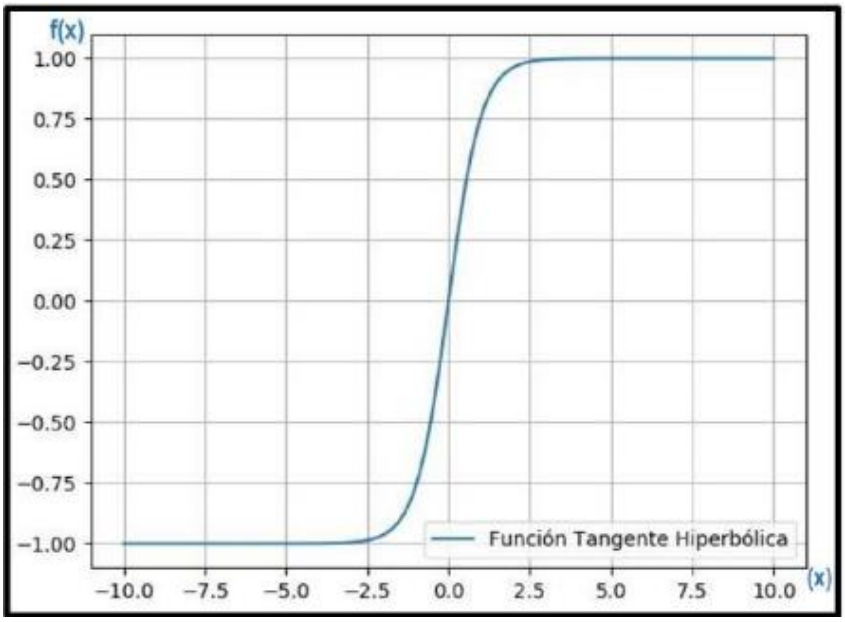
Función sigmoidea



Nota: Adaptado de (Montalvo, 2021).

Figura 11

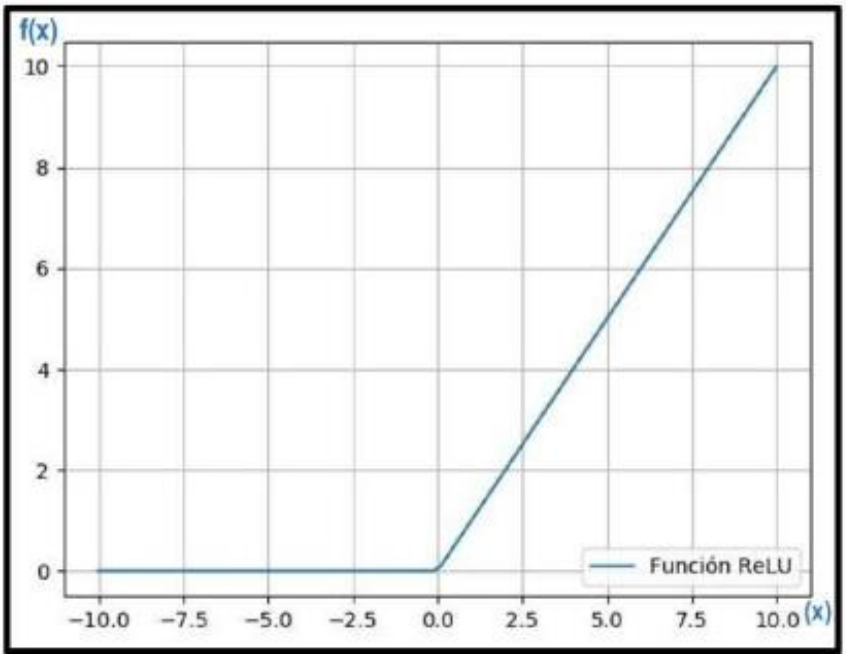
Función tangente hiperbólica



Nota: Adaptado de (Montalvo, 2021).

Figura 12

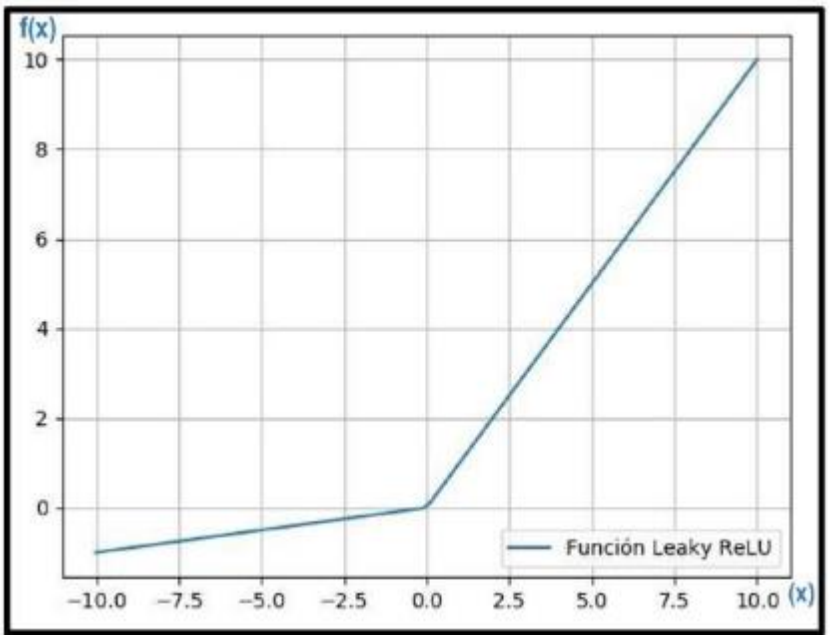
Función ReLU



Nota: Adaptado de (Montalvo, 2021).

Figura 13

Función Leaky ReLU



Nota: Adaptado de (Montalvo, 2021).

2.2.3.3.3 Aprendizaje de las Redes Neuronales Artificiales

El aprendizaje o entrenamiento de redes neuronales ajusta los pesos sinápticos usando datos para que la red pueda realizar la tarea encomendada. Esto se logra minimizando una función de error mediante el algoritmo de gradiente descendiente (Montalvo, 2021).

Este algoritmo encuentra los parámetros que minimizan la 'función de coste', que calcula el error de la red neuronal. Comienza con pesos aleatorios y a través de iteraciones, los pesos se ajustan para reducir el error de la red.

Los conceptos de función de coste y gradiente descendente se explican con más detalle a continuación:

- Función de coste: Una función de costo se define como una medida de la discrepancia entre la predicción del modelo y el valor real, que son los valores conocidos de Y. Esencialmente, la función de costo cuantifica qué tan bien el modelo está estimando la relación entre X e Y. En resumen, la función de costo evalúa el desempeño de un modelo de aprendizaje que emplea redes neuronales artificiales con datos previamente establecidos (Montalvo, 2021).
  Existen varios tipos de funciones de coste. Entre ellas según Montalvo (2021) tenemos la función del error cuadrático medio o *mean square error* (MSE, por sus siglas en inglés). Su fórmula se representa de la siguiente manera:

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(\hat{Y}_n - Y_n)^2 \qquad (7)$$

  Donde: MSE es la función del error cuadrático medio.
  N es el tamaño de la muestra.
  $\hat{Y}_n$ es el vector de N predicciones.
  Y_n es el vector de los verdaderos valores.

  En la función MSE los errores se hacen más grandes debido a que están elevados al cuadrado. Esto trae como consecuencia que la función ralentiza la velocidad de aprendizaje de la red neuronal. Otra de las funciones de coste es la función de entropía cruzada o *cross entropy*. Su fórmula se representa de la siguiente manera:

$$CE = -\frac{1}{N}\sum_{n=1}^{N} Y_n \, log\hat{Y}_n + (1 - Y_n)\log\left(1 - \hat{Y}_n\right) \qquad (8)$$

Donde: CE es la función de entropía cruzada.

N es el tamaño de la muestra.

$\hat{Y}$ es el vector de N predicciones.

Y es el vector de los verdaderos valores.

- Gradiente descendiente: es un algoritmo de optimización eficiente que permite ajustar los pesos de las conexiones entre neuronas para minimizar la función de coste o error (Montalvo, 2021).

  Para ir más allá de los modelos lineales, tendremos que enfrentar que las ecuaciones de pérdida mínima a menudo no tienen soluciones cerradas. En su lugar, se enfrenta a un problema de optimización general en un espacio de pesos continuos. Estos problemas se abordan con un algoritmo de escalada que sigue el gradiente de la función a optimizar. Aquí, se utiliza el descenso de gradiente para minimizar la pérdida. Se elige un punto de partida en el espacio de peso, en este caso, un punto en el plano (0, 1), y luego se mueve a un punto vecino que está cuesta abajo, repitiendo este proceso hasta converger en la pérdida mínima posible (Russell & Norvig, 2010).

  Para cada ω_i en w, se realiza:

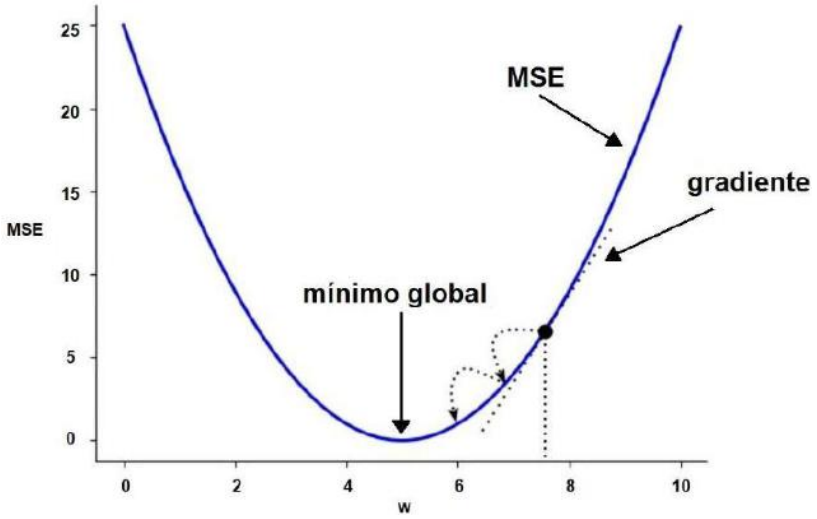
$$\omega_i \leftarrow \omega_i - \alpha \frac{\partial}{\partial \omega_i} Loss(W) \qquad (9)$$

  Donde: W cualquier punto en el espacio de parámetros.

  El parámetro α es la que se denomina tasa de aprendizaje cuando se trata de minimizar la pérdida en un problema de aprendizaje. La tasa de aprendizaje suele ser una constante fija o decrecer con el tiempo a medida que avanza el proceso de aprendizaje. Representamos la visualización del gradiente descendiente en la Figura 14.

Figura 14

Gradiente descendiente



Nota: Fuente (Montalvo, 2021).

A continuación, se presentan diversos métodos de descenso del gradiente según lo expuesto por Russell y Norvig (2010):

- La regla de actualización de Descenso de Gradiente de (lote o *batch*): estas actualizaciones se basan en el algoritmo de descenso de gradiente por lotes para la regresión lineal. Aunque garantiza convergencia al mínimo global con un valor de α pequeño, el proceso es lento ya que recorre todos los datos en cada paso. El descenso de gradiente estocástico, en el que se considera un solo punto de entrenamiento a la vez, es una alternativa más eficiente.
- El descenso de gradiente estocástico: se utiliza en una configuración en línea, donde los nuevos datos ingresan uno a la vez, o fuera de línea, donde se recorren los mismos datos tantas veces como sea necesario, dando un paso después de considerar cada ejemplo individual. A menudo, este método es más rápido que el descenso del gradiente por lotes. Sin embargo, con una tasa de aprendizaje α fija, no garantiza la convergencia; tiende a oscilar alrededor del mínimo sin asentarse.

Los métodos que acabamos de mencionar conforman los cimientos del Aprendizaje Profundo, al permitir la actualización automática de los valores de los pesos. Estos procesos fundamentales son los que posibilitan que los modelos de Aprendizaje Profundo aprendan y mejoren continuamente a partir de los datos disponibles.

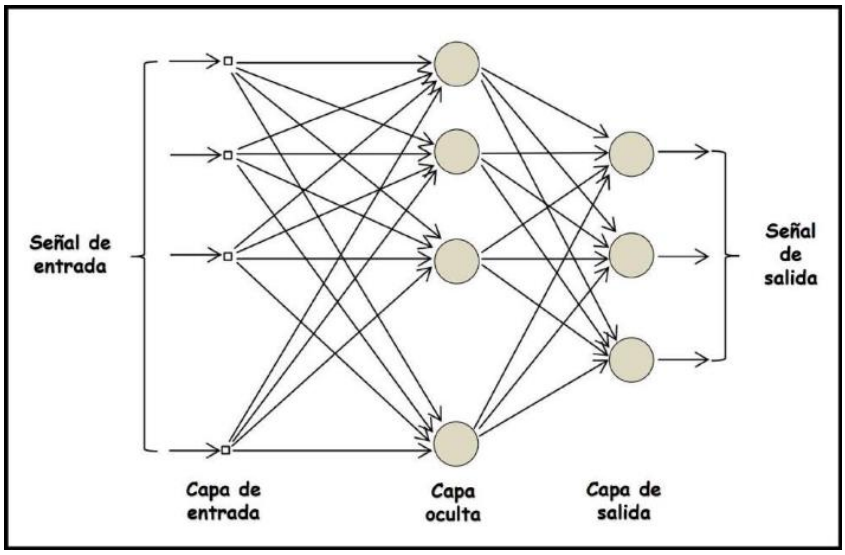
2.2.3.3.4 Construcción de una Red Neuronal Artificial

Las redes neuronales artificiales se clasifican según su topología, arquitectura (Monocapa o Multicapa) y aprendizaje (Supervisado, No supervisado y por Refuerzo). En este caso, por ser el objetivo de esta investigación, solo se mencionarán algunas de las arquitecturas de construcción del tipo de aprendizaje supervisado (Manzanares, 2019).

- Redes con aprendizaje supervisado: tienen una estructura de red neuronal multicapa *feed-forward* como se observa en la Figura 15, con una capa de entrada, una de salida y capas ocultas intermedias. Los pesos y umbrales de las conexiones entre neuronas permiten que la red reconozca patrones, donde la activación de una neurona en una capa activa un conjunto de neuronas en la siguiente capa, formando el patrón detectado Haykin (1999). Este proceso permite que la red aprenda a identificar patrones a partir de los datos de entrenamiento proporcionados.

Figura 15

Estructura de red neuronal feed-forward multicapa



Nota: Adaptado de (Haykin, 1999).

Según Manzanares (2019) las neuronas transmiten señales a las siguientes capas. Cada neurona calcula una suma ponderada de las señales recibidas, y una función de activación determina si se activa o no.

Durante el entrenamiento, se optimizan los pesos para minimizar la pérdida y especializar los patrones de activación.

Características que tienen las neuronas por su tipo de capa Montalvo (2021):

- Neuronas de la capa de entrada: reciben información externa y la transmiten directamente a las neuronas de la capa oculta sin realizar cálculos.
- Neuronas de la capa oculta: no se conectan directamente con el exterior. Realizan los cálculos y transmiten la información de la entrada a la salida.
- Neuronas de la capa de salida: envían información fuera de la red y, a diferencia de las neuronas de entrada, realizan cálculos sobre sus valores de entrada.

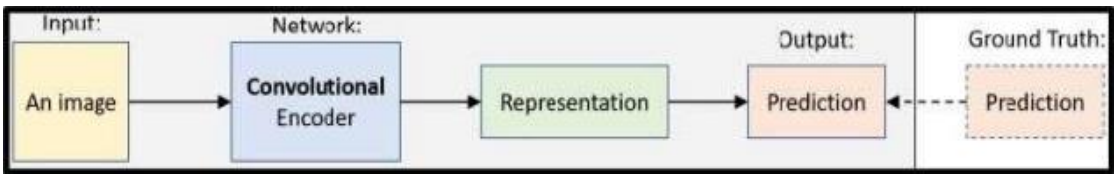
Cada tipo de aprendizaje tiene diferentes paradigmas de arquitectura, con variaciones en los codificadores, entradas y conexiones. Las redes neuronales funcionan como codificadores que detectan patrones en datos para crear representaciones útiles, y como decodificadores que generan nueva información a partir de esas representaciones. En resumen, las redes neuronales codifican datos y los decodificadores generan nueva información (Galdi y Tagliaferri, 2019).

A continuación, veremos la representación de la arquitectura de aprendizaje supervisado que se ajusta mejor a nuestro estudio:

- *Convolutional Neural Networks*: comúnmente conocidas por sus siglas *CNN* (redes neuronales convolucionales) son redes neuronales que utilizan la simetría espacial para aprender de manera eficiente los patrones locales, principalmente en imágenes. La simetría espacial significa que las características de una imagen son similares en diferentes partes de esta. Las CNN comparten los pesos (parámetros) en el espacio para que la detección de las mismas características y patrones sea más eficiente. En lugar de emplear exclusivamente capas densamente conectadas, se recurre a capas convolucionales (codificador convolucional), como se observa en la Figura 16.

Figura 16

Arquitectura Convolutional Neural Networks



Nota: Adaptado de (Salehinejad et al., 2019).

En el siguiente apartado de este acápite hemos previsto profundizar este tipo de arquitectura a detalle.

2.2.3.3.5 Redes Neuronales Convolucionales

Las redes convolucionales según Lecun et al. (1989), también conocidas como Convolutional Neural Networks *(*CNN) o *ConvNet* son un tipo de red neuronal diseñada para procesar datos con estructura de cuadrícula, como series temporales e imágenes.

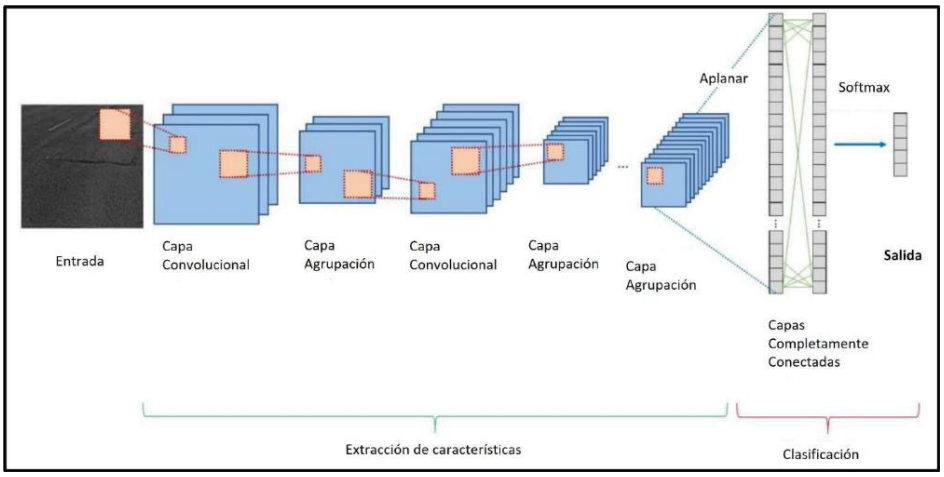
Estas redes utilizan la convolución, en lugar de multiplicación de matrices, al menos en una de sus capas (Goodfellow et al., 2017).

Las CNN son algoritmos de aprendizaje profundo que toman imágenes de entrada y las convoluciona con filtros o núcleos, para extraer características. Una imagen de tamaño $N \times N$ se convoluciona con una matriz de filtro de tamaño $f \times f$, y esta operación de convolución aprende la misma característica en toda la imagen (Chauhan et al., 2018).

Siendo la CNN un modelo de clasificación de imágenes eficiente. Una CNN simple tiene una secuencia de capas que transforman activaciones a través de funciones diferenciables. Consta de tres tipos principales de capas: convolucional, de agrupación y totalmente conectada. Estas capas se apilan para formar la arquitectura CNN, como se muestra en la Figura 17.

Figura 17

Estructura de una red neuronal convolucional (CNN) típica



Nota: Adaptado de (Arya et al., 2021).

Los detalles de las capas se presentan a continuación:

- Capa convolucional (*Convolutional layer*): es la primera capa de una red convolucional. Convierte una imagen de entrada en píxeles y los procesa a través de un filtro, obteniendo así un mapa de características más pequeño y conciso como salida.

En las redes neuronales convolucionales (CNN), cada neurona de una capa está conectada con cada neurona de la siguiente capa, la Ecuación 10 muestra la operación del píxel de salida de la convolución.

$$y_i = \sum_i X_i \times W_{ij} + b_i \qquad (10)$$

Donde: X_i entrada de cada canal.

W_i pesos de cada filtro, controlan conexión entre neuronas.

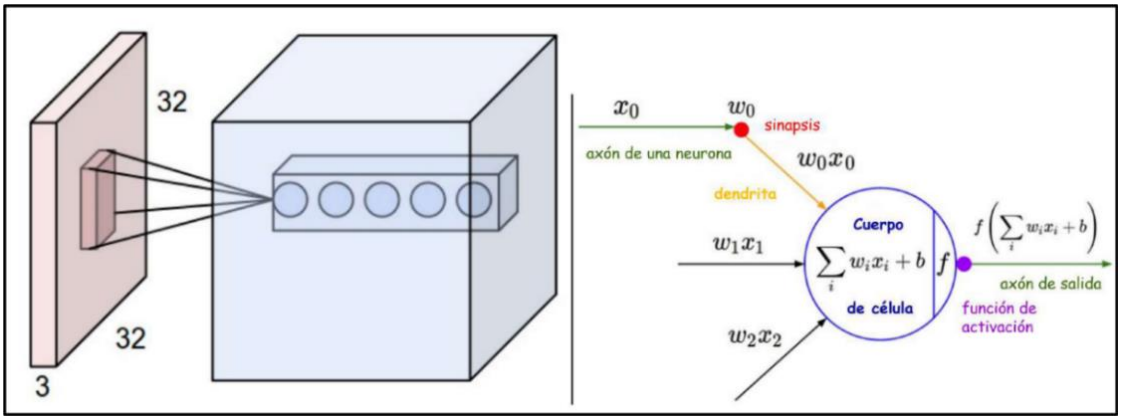
b_i bias, peso adicional en la salida, siempre 1.

y_i píxel de salida.

En la Figura 18 muestra cómo se conectan las neuronas de la capa convolucional al volumen de entrada, según la Ecuación 10. A la izquierda se ve un ejemplo de volumen de entrada, una imagen *RGB* de (32x32x3) y un ejemplo de volumen de neuronas en la capa convolucional.

Figura 18

Ejemplos de volúmenes de entrada y de neuronas en la capa convolucional



Nota: Adaptado de (Universidad Politécnica De Madrid, 2022).

De la figura 18 también se deduce cada neurona en la capa convolucional en azul se conecta solo a una región local en la entrada, pero a través de todos los canales de color. Hay múltiples neuronas (5 en este ejemplo) a lo largo de la profundidad, todas mirando la misma región en la entrada. Estas neuronas comparten el mismo campo receptivo, pero no los mismos pesos.

Es importante tener en cuenta que las neuronas en las capas convolucionales de una CNN calculan un producto escalar de sus pesos con la entrada, seguido de una función de activación no lineal, similar a una red neuronal feed-forward. Sin embargo, estas neuronas están conectadas sólo a una región local de la capa

anterior en el espacio 2D, pero de manera completa a lo largo de la profundidad del volumen de entrada.

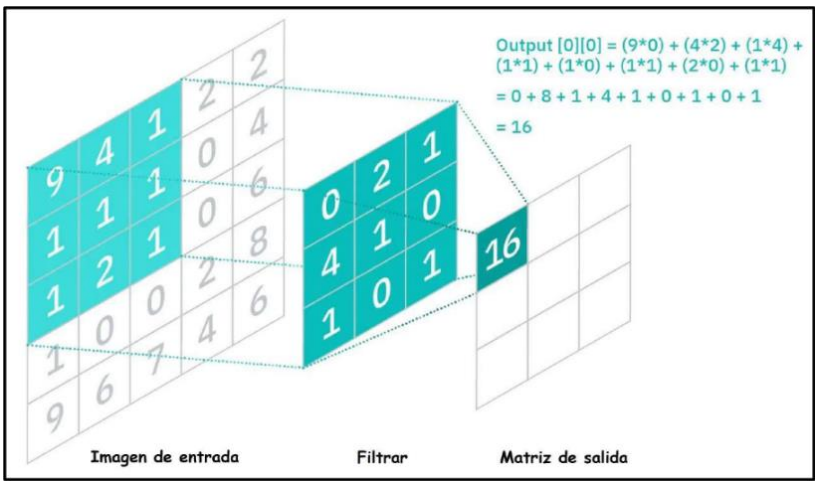
Las redes neuronales convolucionales tienen cuatro hiper-parámetros que se establecen antes de entrenar el modelo. Estos son:

a) Campo receptivo o Tamaño del filtro: se explicó en párrafos anteriores, exactamente de la Figura 18 se refiere al área de entrada a la que cada neurona de la capa convolucional está conectada.

b) *Kernel* o Número de Filtros: es una matriz de números más pequeña que la imagen, que se utiliza en el proceso de convolución para extraer características de la imagen. Calculamos el producto punto del kernel con los valores RGB de cada píxel de la imagen. Recorremos la imagen con el kernel como se aprecia en la Figura 19, obteniendo así la matriz de salida. El kernel determina qué características de la imagen se resaltan, como bordes, desenfoques o enfoques.

Figura 19

Demostración del proceso de convolución

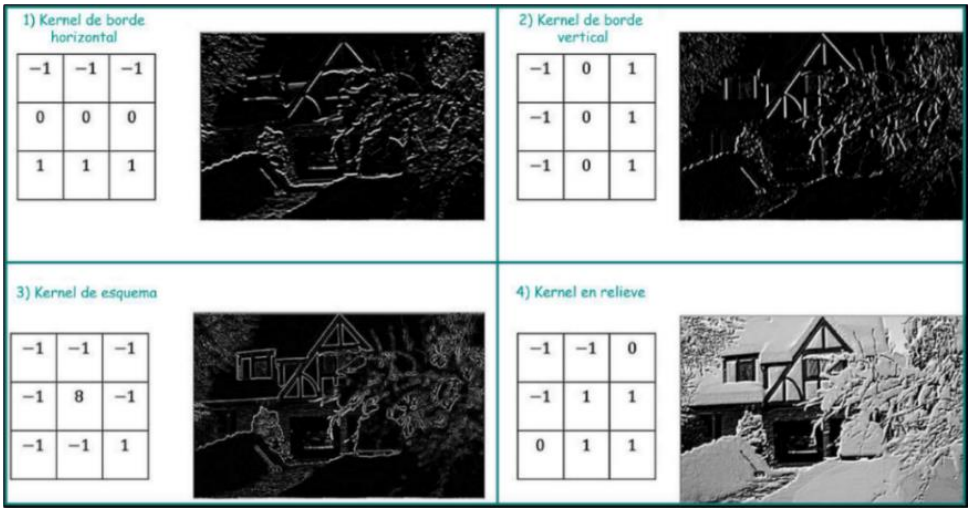


Nota: Adaptado de (Behera, 2021).

En la Figura 20 mostramos las diferencias que hay entre algunos kernels o filtros existentes : en 1) podemos observar de mejor manera la detección de los bordes o líneas que tienen la forma horizontal, así también sucede en 2) las que mejor se logran detectar son los bordes o líneas que tienen la forma vertical, en 3) sobresalen los bordes de las figuras entre líneas formando deferentes tipos de esquemas y en 4) resalta el relieve de los bordes de cada figura que está dentro de la imagen, resultando que se aprecia la diferencia que hay entre cada figura.

Figura 20

Kernels para resaltar diferentes características



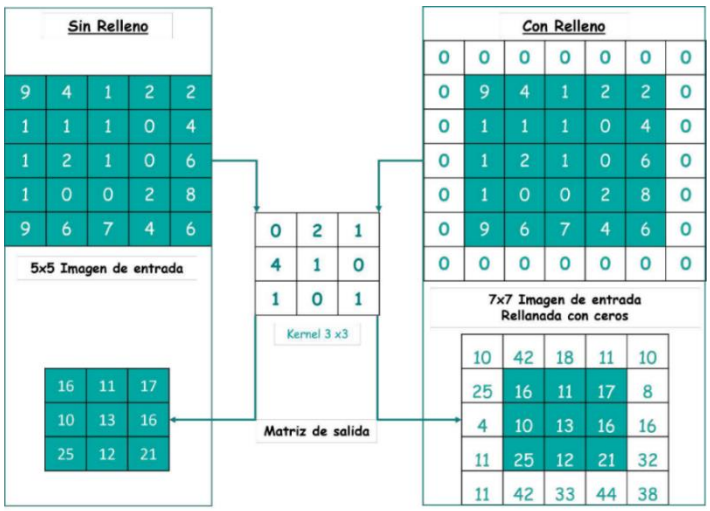
Nota: Adaptado de (Behera, 2021).

c) *Paddings* o Relleno: el padding o relleno es el número de píxeles añadidos junto a cada borde para conservar la forma de la imagen. El relleno se necesita para ayudar en la resolución de la imagen, enfocándose en las características más resaltantes y al mismo tiempo mejora el rendimiento.

Continuando con el ejemplo de la Figura 19 en la Figura 21 podemos observar que al realizar la convolución con un kernel de 3×3 en la imagen de entrada sin relleno y con relleno de ceros obtenemos una matriz de salida con diferente tamaño especial en este caso de los volúmenes de salida.

Figura 21

Convolución de una imagen de entrada sin relleno y con relleno de ceros



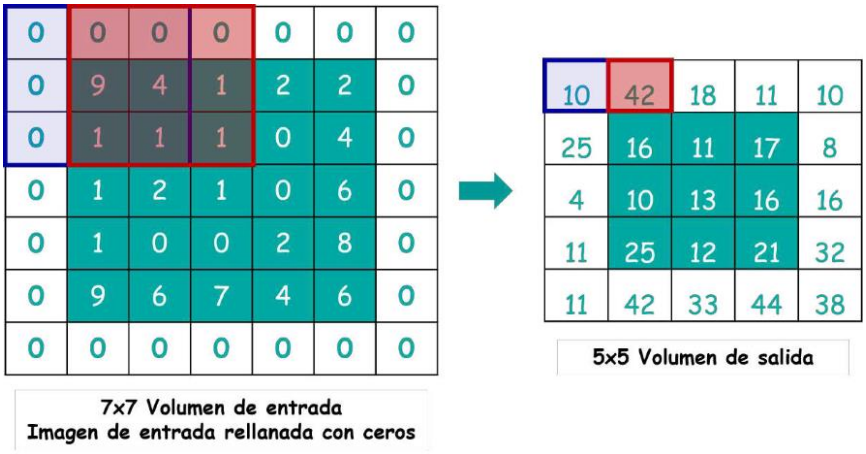
Ahora que ya se ha explicado, deberíamos preguntarnos: ¿Es necesario usar el Paddings o Relleno en el proceso de las CNN?. Mayormente si en las capas CNN no se pusieran ceros en las entradas, sólo se realizarán convoluciones válidas, el tamaño de los volúmenes se reduciría a una pequeña cantidad después de cada convolución, y la información en los bordes se "borraría" demasiado rápido.

d) *Stride* o Zancada: se refiere a la cantidad de píxeles que el kernel salta antes de la siguiente operación de convolución. La dirección de estos pasos depende de la arquitectura del modelo. El valor de la zancada se establece después de fijar los valores de relleno y kernel.

Siguiendo el ejemplo de la Figura 19 en la Figura 22 se muestra cómo se han realizado las zancadas (1,1). Los píxeles se mueven de izquierda a derecha, fila por fila, aplicando la convolución con el volumen de entrada. La primera convolución genera un píxel azul en la imagen de salida, y la segunda convolución, desplazada un píxel a la derecha, genera un píxel rojo.

Figura 22

Zancadas que realiza una imagen de entrada con relleno de ceros



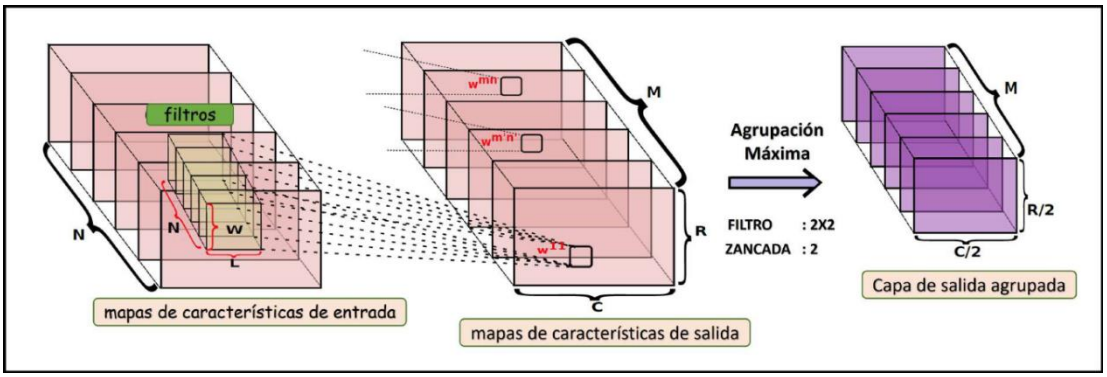
- Capas de agrupación (*Pooling layer*): las capas de agrupación (*pooling*) son importantes en las CNN. Su principal función es lograr invariancia traslacional en el mapa de características. Reemplazan las activaciones en una ventana con una activación representativa, lo que resalta las características detectadas e introduce no linealidad (en el caso de agrupación máxima). Cuando se obtiene el mapa de características después de una convolución, el operador de agrupación (pooling) escanea y agrega información dentro de cada región local. Esta agregación de información hace que la representación sea más robusta a la traducción y distorsión. Por ejemplo, el operador de agrupación promedio o el operador *max*

pooling toman el promedio o el valor máximo de la región local, respectivamente (Moolchandani et al., 2021).

En el ejemplo de la Figura 23, primero se aplican las capas de convolución para detectar características de una imagen RGB. La entrada tiene un tamaño de $R \times C$ con N canales de profundidad, y los filtros también tienen una profundidad de N. Los filtros de tamaño $WxLxN$, también conocidos como núcleos, son matrices de valores o pesos que se convolucionan con la matriz de entrada $RxCxN$, produciendo una salida de tamaño $RxCx1$ (suponiendo que el relleno se hizo adecuadamente). Cuando se aplican M de estos filtros a la entrada, se obtienen M matrices de salida. Estas M matrices de salida se convierten en las matrices de entrada para la siguiente capa, y se les conoce como mapas de características. Estos mapas de características pasan por una capa de max pooling, que toma la activación máxima dentro de una ventana del mapa de características de entrada (ver ejemplo de la Figura 24).

Figura 23

Una capa de convolución y una capa de agrupación en una CNN



Nota: Adaptado de (Moolchandani et al., 2021).

El tamaño de la ventana de activaciones es igual al tamaño del filtro del operador de agrupación, que en este caso es 2x2 con agrupación max pooling. Este operador se desliza en pasos de 2 a través del mapa de características anterior (capa convolucional). Esto reduce las dimensiones del mapa de características, descartando el 75% de los píxeles. Esta operación de muestreo descendente reduce el tamaño de los canales de características, lo que disminuye la carga computacional (Manli et al., 2017).

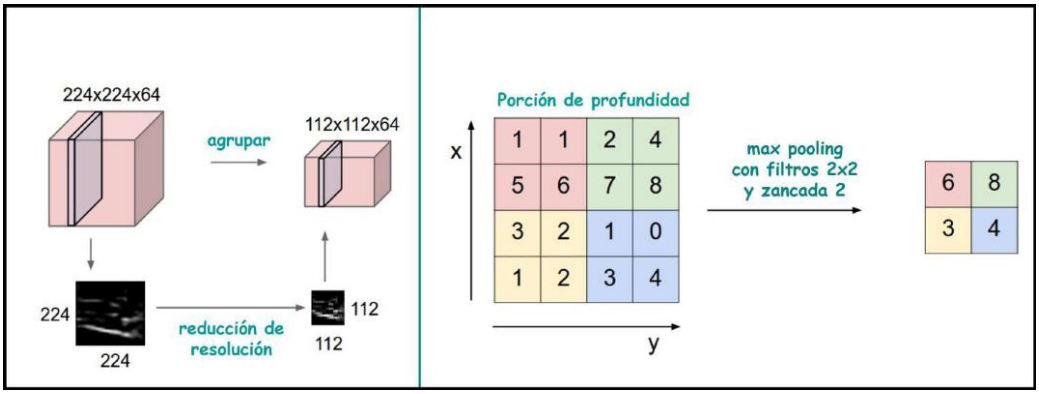
En conclusión, el paso de operador de agrupación y de la operación de muestreo descendente se consideran como un procedimiento de selección de características

que mantiene la información más útil mientras reduce el espacio de funciones. Pero ¿cuándo usar, qué tipo de agrupación? Se conoce que la agrupación promedio selecciona los valores promedio y por lo tanto escoge las características de la imagen que se suavizan y no se detectan características más nítidas, mientras que para la agrupación máxima se seleccionan los valores más altos y, por lo tanto, los colores más brillantes se detectan fácilmente.

La Figura 24 muestra cómo se reduce la resolución usando max pooling. En el ejemplo de la izquierda, un volumen de $[224 \times 224 \times 64]$ se procesa con un filtro de 2 y paso 2, generando un volumen de $[112 \times 112 \times 64]$. Esto reduce la resolución, pero mantiene la profundidad. En el ejemplo de la derecha, max pooling con paso 2 selecciona el valor máximo de cada matriz $[2 \times 2]$), eliminando los píxeles con valores bajos o cero que no contienen características útiles.

Figura 24

La capa de agrupación reduce el espacio de la muestra.



Nota: Adaptado de (Fei, s.f.).

- Capas completamente conectadas (*Fully connected layers*): La capa totalmente conectada es generalmente la última capa en la estructura de CNN. Cada neurona usa una función de activación (generalmente, la función sigmoide). En este caso, se utiliza ReLU, que es una función lineal (presentada en el acápite 2.2.3.3.2), la cual está completamente vinculada a las neuronas de la capa anterior, generando la entrada directamente si es positiva; de lo contrario, generará cero. La capa totalmente conectada integra información local, tiene la capacidad de discriminar clases, y la salida de las neuronas se pasa a la capa de salida. Por lo tanto, la capa completamente conectada tiene algún papel de clasificadores convencionales. Si la capa h es la capa completamente conectada, la salida de esta capa estará compuesta por la Ecuación 11:

$$a_n^l = f(W^l . a_n^l - 1 + b^l) \qquad (11)$$

Donde: W^l es el núcleo convolucional.

b^l es el termino de desplazamiento.

La siguiente capa de perceptrones aplica la activación softmax (Ecuación 12), una función logística generalizada utilizada en la clasificación multiclase. La salida de esta capa corresponde a las probabilidades de clasificación, cuya suma es 1. La clasificación con la probabilidad máxima se selecciona como la salida final. La función softmax se representa en la Figura 25.

$$a_n^l = O_n^v = softmax(W^l . a_n^l - 1 + b^l) \qquad (12)$$

Donde: W^l es el núcleo convolucional.

b^l es el termino de desplazamiento.

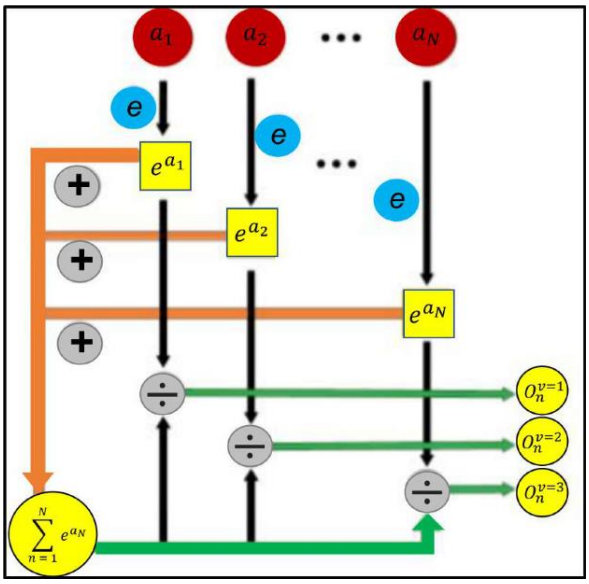
La fórmula de error E_n en la enésima muestra se representa en la ecuación 13.

$$E_n = \frac{1}{2}\sum_{v=1}^{V} \|t_n^v - O_v^n\| \qquad (13)$$

Donde: la probabilidad de salida esperada de la enésima muestra en v categorías se representa como t_n^v. El error global de N muestras se calcula utilizando la ecuación.

Figura 25

Proceso de clasificación softmax



Nota: Adaptado de (Ros, 2019).

Y para finalizar tenemos la Ecuación 14 que se expresa a continuación:

$$E = \sum_{n=1}^{V} E_n \qquad (14)$$

Donde: t_n^v es la probabilidad esperada de la muestra enésima en v categorías de clasificación.

Según los análisis, la capa totalmente conectada y la salida podrían ser equivalentes. Cuando se entrena una red CNN con pocos datos, es probable que se produzca un sobreajuste. La técnica de abandono (dropout) desactiva aleatoriamente algunas neuronas durante la propagación en la CNN para evitar el sobreajuste. Después de varias iteraciones, se alcanza la pérdida mínima, y el aprendizaje de las neuronas es más robusto, lo que indica que el modelo está entrenado para la clasificación (Ros, 2019).

2.2.4 Entrenamiento de CNN para detectar grietas en superficie de pavimento.

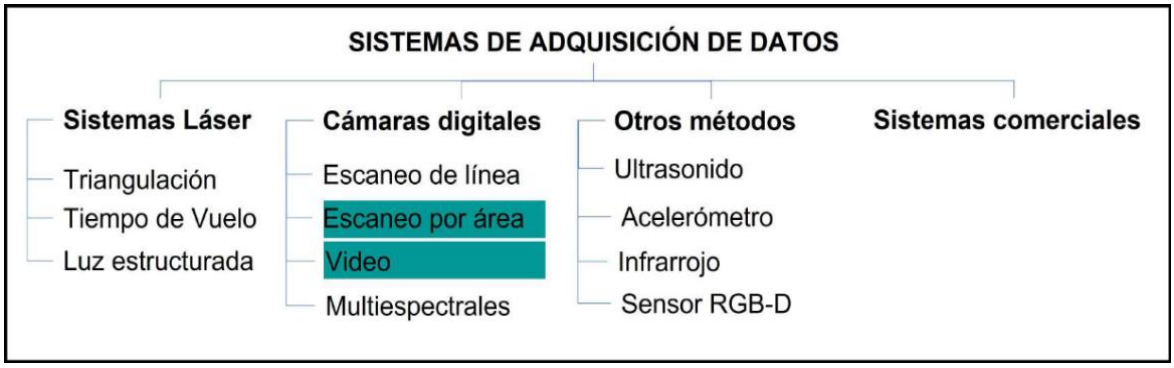
Para entrenar un modelo de CNN para detectar grietas en la superficie del pavimento, se requieren una serie de pasos preparatorios que permitirán alcanzar dicho objetivo. Estos pasos incluyen:

2.2.4.1 Sistema de adquisición de datos

Según Ríos et al. (2020), los sistemas utilizados para adquirir los datos de la superficie del pavimento se dividen en cuatro grupos. Estos sistemas de adquisición se observan en la Figura 26, pero el enfoque se centrará en dos que pertenecen al grupo de cámaras digitales.

Figura 26

Sistemas de adquisición de datos para detección de fallas superficiales



Nota: Adaptado de (Ríos et al., 2020).

Los sistemas de escaneo por área y video utilizan cámaras de teléfonos inteligentes. Estos dispositivos tienen sensores de matriz de píxeles que permiten generar imágenes bidimensionales en un solo ciclo de exposición. Los teléfonos de gama alta cuentan con las mejores cámaras y la opción de grabar video, cuya duración depende del almacenamiento del dispositivo.

Además, los teléfonos inteligentes tienen cámaras que se integran en una plataforma que permite usar funciones sin necesitar equipos externos, como acceso a 5G, GPS, Bluetooth, WiFi, entre otras ventajas que ofrece la interfaz del smartphone.

En este caso, el equipo de smartphone de gama alta utilizado en esta investigación es el Xiaomi 11T Pro, que cuenta con las siguientes características:

- Procesador Snapdragon 888, 12GB RAM y 256GB de almacenamiento.
- Cámara principal de 108MP, con cámaras secundarias de 8MP y 2MP.
- Batería de 5000 mAh con carga rápida *HyperCharge* de 120W, que se carga completamente en 17 minutos.
- Software de inteligencia artificial en la cámara que detecta objetos, condiciones de luz y movimiento, y ajusta los parámetros automáticamente.

Figura 27

Parte trasera del equipo smartphone de gama alta Xiaomi 11T Pro

2.2.4.2 Preparación de la base de datos sobre el mal estado del pavimento

Disponer de datos suficientes es la piedra angular del uso del aprendizaje profundo, para detectar el deterioro de los pavimentos. Habiendo examinado y analizado diferentes estudios sobre la preparación de un conjunto de datos del mal estado del pavimento, tenemos claro que se debe tener en cuenta los conceptos que se mencionan a continuación:

2.2.4.2.1 Recopilación del conjunto de datos

El primer paso y el más importante para entrenar una red neuronal es crear el conjunto de datos del que aprenderá. Se debe tener en cuenta que la calidad del conjunto de datos en sí tendrá una gran influencia en la precisión final de la red. Según Montalvo (2021), se debe alimentar la red con ejemplos de los objetos de interés desde todas las perspectivas.

Las imágenes suelen ser recolectadas a partir distintas alternativas y fuentes, estas son:

- El conjunto de datos se obtiene mediante el uso de un sistema de adquisición de datos, como se mencionó en el acápite 2.2.4.1.
- Una segunda opción sería descargar de la web un conjunto de datos de imágenes del deterioro del pavimento de otros investigadores, en la Tabla 2 mencionamos algunas de las investigaciones que tienen un conjunto de datos de daños en pavimentos.

Tabla 2

Un resumen de los últimos conjuntos de datos de imágenes viales existentes

Conjunto de datos	Área de estudio	Cantidad imágenes	Resolución	Dispositivo utilizado
Majidifard et al. (2020)	Estados Unidos	7237	640 x 640	API de Google
Maeda et al. (2018)	Japón	9053	600 x 600	Smartphone: LG Nexus 5X
Angulo et al. (2019)	Japón, México y Italia	18,034	600 x 600	Smartphone
Arya et al. (2021)	India, Japón y República Checa	26,620	600 x 600, 720 x 960	Smartphone: LG Nexus 5X y Samsung Galaxy J6

Nota: Adaptado de (Arya et al., 2021).

- Una tercera opción sería descargar individualmente desde Internet las imágenes y videos que contengan los objetos. Por ejemplo, utilizando un motor de búsqueda como Google.
- Cuarta opción sería obtener imágenes de *Google Street View* (*GSV*) permite visualizar fotografías panorámicas. Por ejemplo, el conjunto de datos que se recopiló en Maniat et al. (2021) investiga la utilidad de utilizar GSV para evaluar la calidad del pavimento.

Las alternativas mencionadas se deciden a partir de los recursos disponibles y también dependen de los objetivos que se quieren alcanzar. En algunos estudios, como el de Arya et al. (2021), se logra combinar y agrupar datos de distintas fuentes.

2.2.4.2.2 Aumento del conjunto de datos.

Aumentar la cantidad de datos de entrenamiento mediante transformaciones de las imágenes originales se conoce como data augmentation. Esto es importante para el entrenamiento de redes neuronales convolucionales, que requieren un conjunto de datos de imágenes suficientemente grande.

La aumentación de datos es un método común para reducir el sobreajuste, que ocurre cuando un modelo se ajusta demasiado a los datos de entrenamiento, impidiendo que generalice bien a nuevos datos. Esto es similar a cuando las personas aprenden de memoria sin entender el concepto (Ortega, 2021).

Según Ortega (2021), las transformaciones más utilizadas son las siguientes:

- Voltear: el usuario gira las imágenes de forma horizontal y vertical. Esta opción permite invertir las imágenes en ambos ejes.
- Rotar: el usuario gira la imagen según un ángulo dado. Los ángulos de rotación más comunes son 90, 180 y 270 grados. La opción permite ajustar la imagen a ángulos predefinidos.
- Escalar: esta opción permite redimensionar la imagen, tanto ampliándola como reduciéndola. El usuario ajusta el tamaño de la imagen según sus necesidades, expandiendo o contrayendo la misma.
- Traslación: la opción permite trasladar la imagen a lo largo del eje X o Y, o en ambos. El usuario desplaza la imagen a su ubicación deseada en el espacio bidimensional.
- Filtros: el usuario cuenta con la capacidad de seleccionar y aplicar efectos visuales a la imagen, brindándole así un mayor control creativo sobre el resultado final.

2.2.4.2.3 Anotación de imágenes

Según Everingham et al. (2015) con *PASCAL VOC*, y Lin et al. (2014) con *Microsoft COCO*, la clasificación, la detección y la segmentación son tres métodos diferentes de etiquetado de datos.

Para la clasificación de imágenes, la anotación se hace a nivel de imagen, requiriendo una etiqueta binaria que indique si el objeto aparece o no. En el caso de la clasificación de daños en el pavimento, un enfoque común es dividir la imagen en sub-imágenes, lo que se conoce como clasificación a nivel de parche.

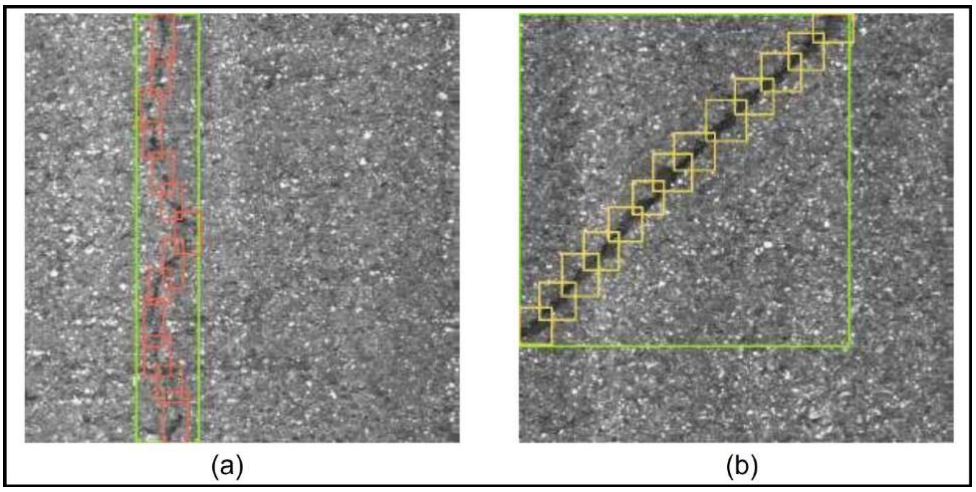
En la detección de objetos, la anotación se hace a nivel de objeto, indicando la categoría y la posición del objeto mediante un cuadro delimitador.

Para la segmentación, la anotación se hace a nivel de píxel, donde cada píxel de la imagen se etiqueta con la categoría o como fondo. La segmentación de imágenes permite generar una localización y estructuras de grietas precisas. Esta funcionalidad posibilita clasificar diferentes tipos de grietas y evaluar características como longitud y ancho. No obstante, este método exige mayor potencia de cómputo en comparación con los otros dos enfoques mencionados anteriormente.

Yang et al. (2022) propone usar anotaciones densas y redundantes en lugar de anotaciones a nivel de parche. Estas anotaciones muestran la ubicación y estructura de las grietas de manera más precisa que las anotaciones tradicionales, como se observa en la Figura 28. En (a), la grieta longitudinal se etiqueta con mayor precisión usando anotaciones densas y redundantes (rojo) en comparación con la anotación tradicional (verde). En (b), la grieta sellada con una pendiente de 45° también se etiqueta con mayor precisión mediante las anotaciones densas y redundantes (amarillo) en comparación con la anotación tradicional (verde).

Figura 28

Comparación de la anotación tradicional y el método de anotaciones densas



Nota: Adaptado de (Yang et al., 2022).

2.2.4.3 Diseño de Red Neuronal Base

En el diseño de la estructura de red neuronal base en el ámbito del aprendizaje profundo, existe la posibilidad de emplear la técnica de transferencia de aprendizaje (*Transfer Learning*) y modelos pre-entrenados.

2.2.4.3.1 Transfer Learning

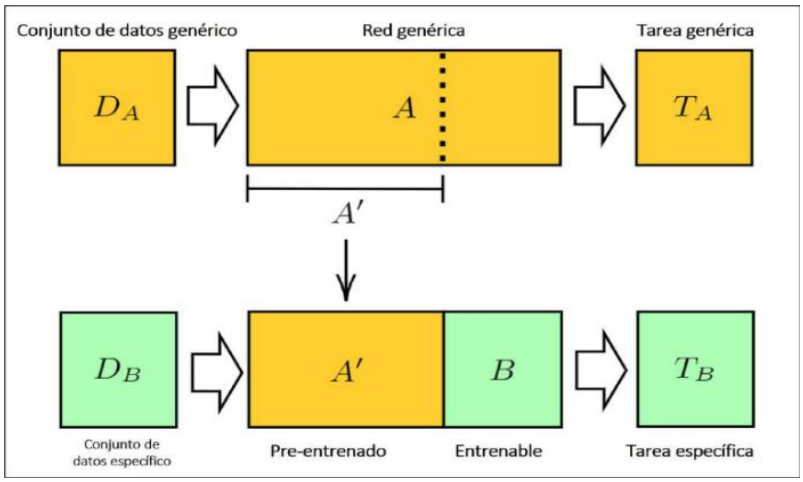
Entrenar un modelo de aprendizaje profundo para detectar un objeto específico no es fácil. Se necesita tener acceso a gran cantidad de datos de entrenamiento de calidad, mucho tiempo y mucha capacidad de cómputo para procesar miles de millones de imágenes y ajustar los pesos de cada clase. Sin embargo, es posible entrenar una red para reconocer nuevas clases a partir de pesos previamente entrenados para otras clases. Este proceso se conoce como Transferencia de Aprendizaje o Transfer Learning (*TL*).

El TL ofrece grandes beneficios en tiempo y precisión. Es una técnica de Aprendizaje Profundo que usa modelos de CNN entrenados previamente para clasificar otros tipos de imágenes. Es útil cuando no se tiene una gran base de datos, ya que recolectar imágenes suele ser una tarea costosa y difícil (Ortega, 2021).

La transferencia de aprendizaje consiste en utilizar una red neuronal previamente entrenada (A') y ajustar sus pesos para reconocer nuevas clases. Esto se logra entrenando las nuevas capas de la red con un conjunto de datos específicos (B) para mejorar la precisión en la tarea a predecir, como se muestra en la Figura 29.

Figura 29

Transfer Learning de Red pre-entrenada en base a un conjunto de datos



Nota: Adaptado de (Sensio, 2020).

En resumen, el Transfer Learning en Aprendizaje Profundo utiliza redes neuronales pre-entrenadas. Las estrategias son:

- Utilizar modelos pre-entrenados como extractores de características (estrategia 1): reutilizar una red pre-entrenada sin la capa final para otra tarea.
- Ajustar modelos pre-entrenados (estrategia 2): reemplazar la última capa y reentrenar selectivamente otras capas, fijando el peso de algunas durante el entrenamiento para mejorar la precisión.

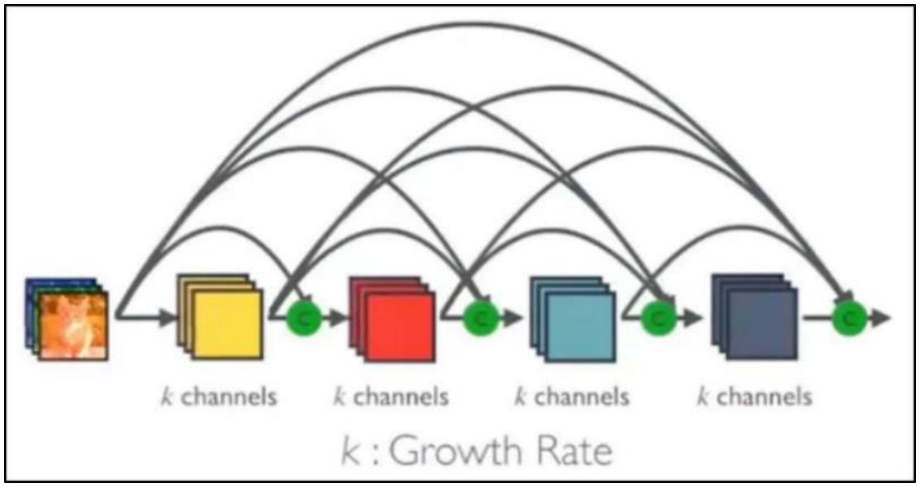
2.2.4.3.2 Modelos de detección

Uno de los requisitos fundamentales del aprendizaje por transferencia es la presencia de modelos que funcionen bien para las tareas específicas a las que se le designe. Existen varias arquitecturas de Aprendizaje Profundo, en esta oportunidad solo se presentarán aquellas arquitecturas de redes neuronales pre-entrenadas que son de interés de esta investigación, y estas son:

- Arquitectura DenseNet: se caracteriza por la conexión de cada capa con todas las anteriores, usando concatenación, lo que permite el flujo de características a través de la red. En la Figura 30 se aprecia que cada capa recibe entradas de todas las capas precedentes, y la tasa de crecimiento K define el número adicional de canales por capa.

Figura 30

Bloque denso en DenseNet con tasa de crecimiento k

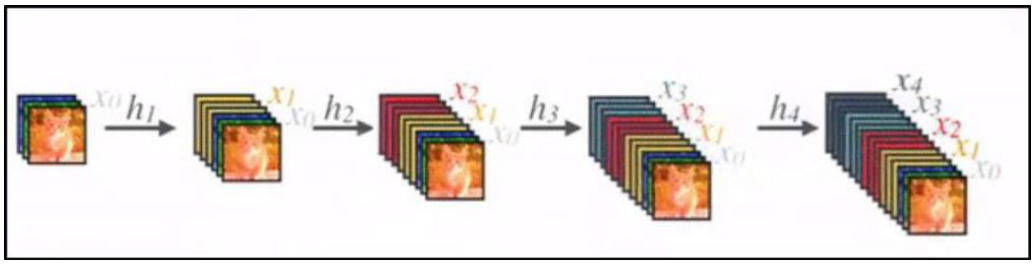


Nota: Adaptado de (Tsang, 2018).

Por lo tanto, se tiene una mayor eficiencia computacional y eficiencia de memoria. La Figura 31 muestra el concepto de concatenación durante la propagación directa:

Figura 31

Concatenación durante la propagación hacia adelante

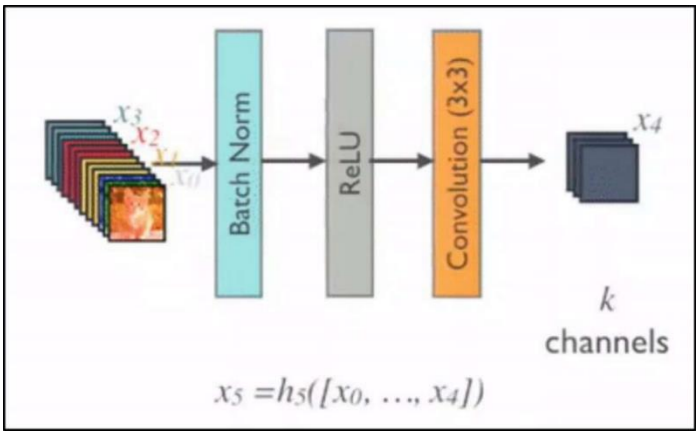


Nota: Adaptado de (Tsang, 2018).

Presentamos en la Figura 32 las capas de composición de una arquitectura DenseNet básica consta de *Batch Norm (BN)*, ReLU y una convolución 3×3 que transforma mapas de características de entrada a mapas de salida de k canales, digamos, siguiendo la idea de *Pre-Activation ResNet*.

Figura 32

Capa de composición DenseNet básico

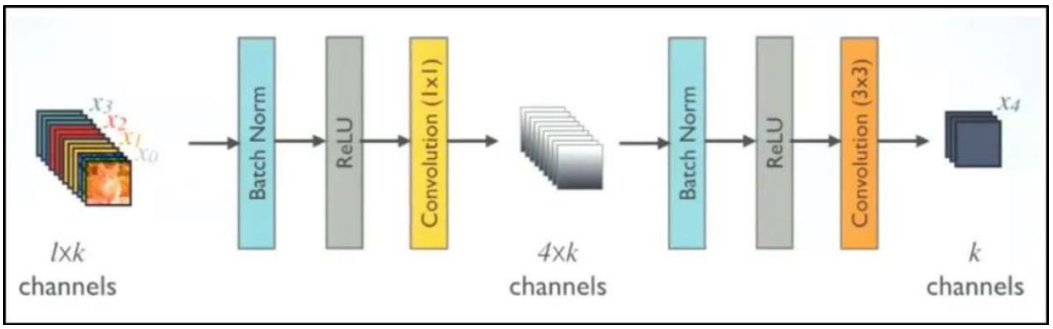


Nota: Adaptado de (Tsang, 2018).

La DenseNet-B, o capas de cuello de botella, reduce la complejidad y el tamaño del modelo. La Figura 33 muestra DenseNets regulares que usan convoluciones 1×1 para disminuir el tamaño de los mapas de características antes de las convoluciones 3×3, mejorando así la eficiencia. La “B” se refiere a la capa de cuello de botella, similar a las usadas en ResNets.

Figura 33

Capa de composición DenseNet-B



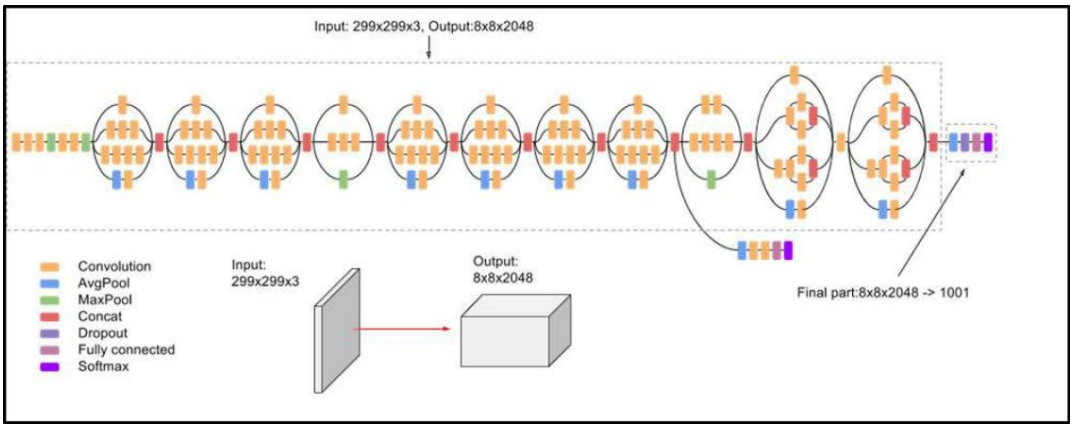
Nota: Adaptado de (Tsang, 2018).

DenseNets-C es una versión de DenseNets-B que reduce la cantidad de características de salida. Se conoce como DenseNet-BC y utiliza un factor de compresión $0 < \theta \leq 1$ para determinar el número de características de salida a partir de los m mapas de características del bloque denso.

- Arquitectura *Inception V3*: es un modelo de clasificación de imágenes basado en redes neuronales convolucionales. El modelo de Inception V3 consta de 42 capas, más que los modelos anteriores *Inception V1* y *V2*, y se muestra en la Figura 34. Su eficiencia es impresionante. La Tabla 3 detalla los componentes que forman este modelo.

Figura 34

Composición de capas de Modelo inception v3



Nota: Adaptado de (Szegedy et al., 2016).

Tabla 3

Composición del Modelo Inception V3

TYPE	PATCH / STRIDE SIZE	INPUT SIZE
Convolution	$3 \times 3/2$	$299 \times 299 \times 3$
Convolution	$3 \times 3/1$	$149 \times 149 \times 32$
Convolution Padded	$3 \times 3/1$	$147 \times 147 \times 32$
Pool	$3 \times 3/2$	$147 \times 147 \times 64$
Convolution	$3 \times 3/1$	$73 \times 173 \times 64$
Convolution	$3 \times 3/2$	$71 \times 71 \times 80$
Convolution	$3 \times 3/1$	$35 \times 35 \times 192$
3 x Inception	Module 1	$35 \times 35 \times 288$
5 x Inception	Module 2	$17 \times 17 \times 768$
2 x Inception	Module 3	$8 \times 8 \times 1280$
Pool	8×8	$8 \times 8 \times 2048$
Linear	Logits	$1 \times 1 \times 2048$
Softmax	Classifíer	$1 \times 1 \times 1000$

Nota: Adaptado de (Szegedy et al., 2016).

- Arquitectura *Mobilenet V2:* este modelo de arquitectura se basa en las ideas de *Mobile Net V1* que según Howard et al. (2017) es una arquitectura optimizada V2 que introduce dos nuevas características a la arquitectura: primero inserta capas de cuello de botellas lineales en los bloques convolucionales porqué la evidencia experimental sugiere que el uso de capas lineales es crucial, ya que evita a las no lineales de destruir demasiada información en un espacio de baja dimensión. Y segundo las conexiones de acceso directo (residuales) que conectan los cuellos de botella funcionan mejor que los atajos que conectan las capas extendidas (Sandler et al., 2018). La estructura detallada de este bloque se detalla en la Tabla 4.

Tabla 4

Bloque residual, se transforma de k a K' canales.

Input	Operator	Output
$h \times \omega \times k$	1×1 conv2d, ReLU6	$h \times \omega \times tk$
$h \times \omega \times tk$	3×3 dwise s=s, ReLU6	$\frac{h}{s} x \frac{\omega}{s} x \frac{t}{k}$
$\frac{h}{s} \times \frac{\omega}{s} \times tk$	linear 1×1 conv2d	$\frac{h}{s} x \frac{\omega}{s} x\, tk'$

Nota: Adaptado de (Sandler et al., 2018).

Con la excepción de la primera capa, usamos una tasa de expansión constante en toda la red. Lo recomendable es utilizar un factor de expansión de 6 aplicado al tamaño del tensor de entrada.

La Tabla 5 muestra secuencias de una o más capas idénticas repetidas, con el mismo número de canales de salida. La primera capa de cada secuencia tiene un paso, las demás usan paso 1. Todas las circunvoluciones espaciales usan núcleos de 3×3. Todas las circunvoluciones espaciales usan núcleos de 3×3.

El factor de expansión siempre se aplica al tamaño de entrada como se describe en la Tabla 4.

Tabla 5

Composición de las capas de MobileNet V2

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1×1	-	1280	1	1
$7^2 \times 1280$	avgpool 7×7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1×1	-	*k*	-	

Nota: Adaptado de (Sandler et al., 2018).

Para terminar en la Tabla 6 se detalla el número máximo de canales/memoria (en Kb) que debe materializarse en cada resolución espacial para diferentes arquitecturas. Asumimos flotantes de 16 *bits* para activaciones. Para *ShuffleNet*, usamos $2x, g = 3$ que coincide con el rendimiento de MobileNetV1 y MobileNetV2.

Para la primera capa de MobileNetV2 podemos emplear el atajo de las capas de cuello de botella residuales invertidas que permiten reducir el requisito de la memoria.

Tabla 6

Tamaños necesarios en cada resolución entre MobileNet V1, V2 y Shuffle Net

Size	MobileNet V1	MobileNet V2	ShuffleNet ($2x$, g=3)
112 × 112	64/1600	16/400	32/800
56 × 56	128/800	32/200	48/300
28 × 28	256/400	64/100	400/600 K
14 × 14	512/200	160/62	800/310
7 × 7	1024/199	320/32	1600/156
1 × 1	1024/2	1280/2	1600/3
max	1600 K	400 K	600 K

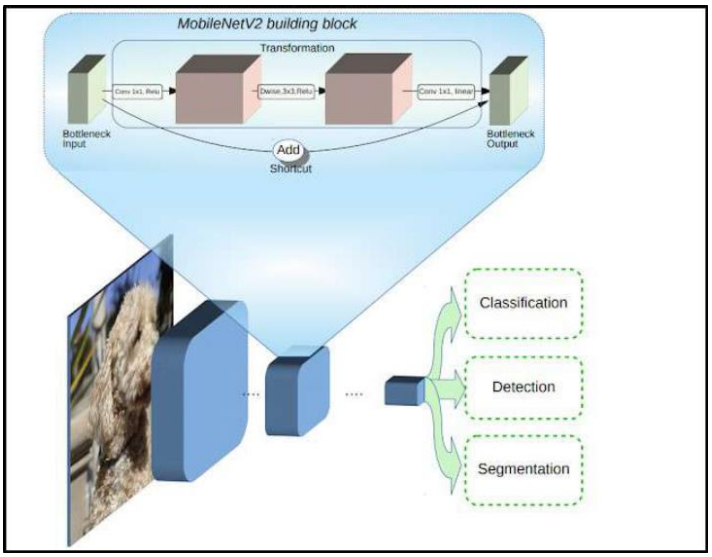
Nota: Adaptado de (Sandler et al., 2018).

En resumen, en la Figura 35 se aprecia una descripción general del modelo MobileNet V2 detallado en las Tablas 4, 5 y 6. En este proceso, los cuellos de botella codifican las entradas y salidas intermedias del modelo, mientras que la capa interna convierte conceptos de nivel inferior, como píxeles, en descriptores de nivel superior, como categorías de imágenes.

Finalmente, los atajos permiten un entrenamiento más rápido y una mayor precisión según Sandler et al. (2018).

Figura 35

Ilustración general de la arquitectura MobileNetV2



Nota: Adaptado de (Szegedy et al., 2016).

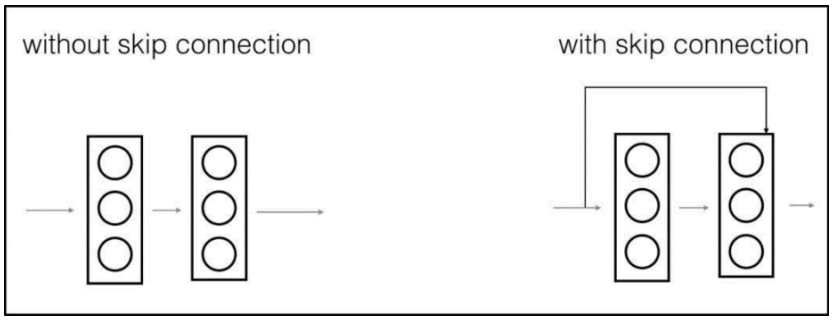
- Arquitectura *ResNet 50*: en los últimos años, las redes neuronales se han vuelto más profundas. La razón principal de esto es que las redes más profundas aprenden una mayor variedad de funciones, desde cosas simples como bordes hasta funciones muy complejas, en el caso de tareas como el reconocimiento de imágenes. Sin embargo, la complejidad de estas redes profundas también trae problemas. Un gran desafío es el fenómeno de los gradientes que desaparecen. Cuando se propaga el gradiente hacia atrás a través de muchas capas, el valor del gradiente se vuelve extremadamente pequeño, lo cual dificulta mucho el entrenamiento de estas redes neuronales enormes.

  Los trabajos realizados por He et al. (2016) en la arquitectura ResNet ha desarrollado un marco de aprendizaje residual que facilita el entrenamiento de redes neuronales muy profundas, reformulando las capas como funciones residuales en lugar de aprender funciones sin referencia a las entradas.

  ResNet introdujo las conexiones de salto, permitiendo que el gradiente se propague hacia capas anteriores. La Figura 36 muestra capas de convolución apiladas a la izquierda y una conexión directa desde la entrada a la salida del bloque de convolución a la derecha, conocida como conexión de omisión.

Figura 36

Omitir conexiones en la arquitectura ResNet

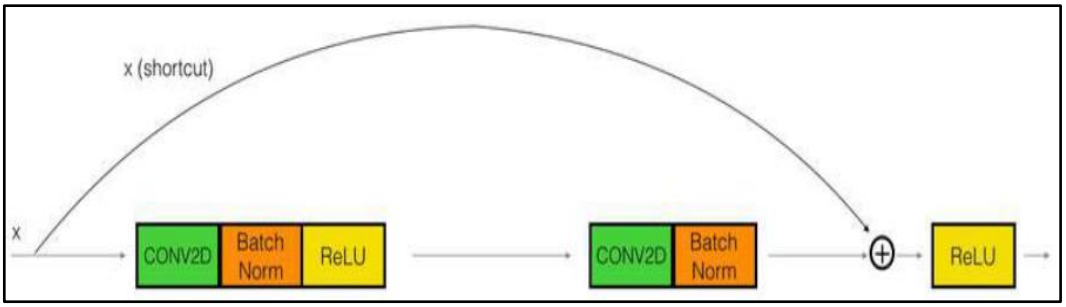


Nota: Adaptado de (Dwivedi, 2019).

Hay dos tipos principales de bloques que se utilizan en una ResNet, dependiendo principalmente de si las dimensiones de entrada/salida son iguales o diferentes. La primera es el bloque de identidad que se ilustra en la Figura 37, es un bloque estándar utilizado en ResNets que corresponde al caso donde la activación de entrada tiene la misma dimensión que la activación de salida. Esto quiere decir, que sucede cuando x y x (*shortcut*) tienen la misma forma.

Figura 37

Bloque de identidad

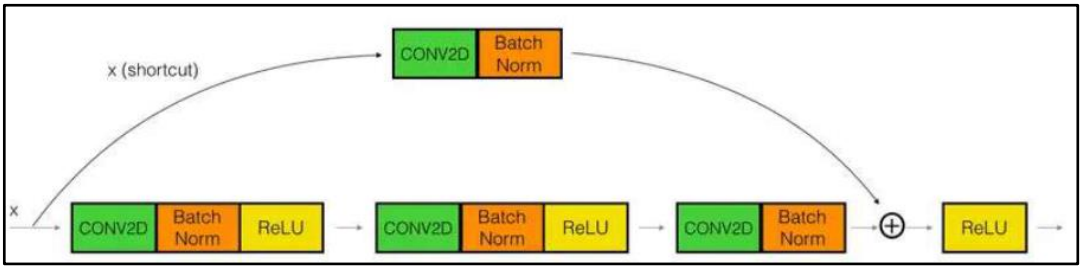


Nota: Adaptado de (Dwivedi, 2019).

Y segundo se presenta en la Figura 38 el bloque convolucional que es un tipo de bloque que se usa cuando las dimensiones de entrada y salida no coinciden. Esto permite x (shortcut) pase por el bloque de convolución elegida de modo que la salida tenga la misma dimensión que la salida del bloque de convolución.

Figura 38

Bloque de Convolución

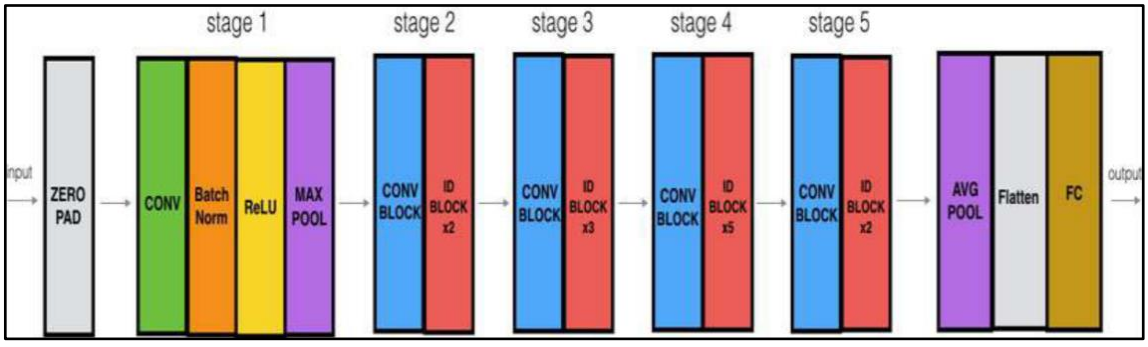


Nota: Adaptado de (Dwivedi, 2019).

Se ilustra de manera general en la Figura 39 el modelo ResNet-50 que consta de 5 etapas, cada una con un bloque de convolución e identidad.

Figura 39

Modelo de ResNet-50



Nota: Adaptado de (Dwivedi, 2019).

Cada bloque de convolución tiene 3 capas de convolución y cada bloque de identidad también tiene 3 capas de convolución. ResNet-50 tiene más de 23 millones de parámetros entrenables.

2.2.4.4 Entrenamiento de una CNN

Una red neuronal convolucional es un conjunto de neuronas interconectadas que extrae las características clave de las imágenes. Se basa en combinar parámetros para predecir resultados. Entonces, ¿cuál es la combinación correcta de estos parámetros? Aquí es donde el proceso de entrenamiento resulta fundamental, ya que implica probar diferentes combinaciones de parámetros para reducir al mínimo el error de predicción.

El entrenamiento de una red neuronal convolucional con conjuntos de datos grandes es más eficiente en GPU que en CPU, ya que las GPU pueden procesar las imágenes en paralelo, lo que acelera enormemente el proceso. Mientras que el entrenamiento en CPU puede tomar días, en una GPU solo lleva horas o minutos. Y esto lo afirman muchos de los trabajos de investigación de detección de fallas en pavimentos flexibles que utilizan técnicas de procesamiento de aprendizaje profundo, como, por ejemplo: en Eisenbach et al. (2017) afirma que en los “experimentos de regularización, estructura de red y todas las demás evaluaciones de parámetros se llevaron a cabo durante aproximadamente tres meses en dos GPU *NVIDIA Titan X*. El tiempo promedio para entrenar un modelo en una sola GPU fue de 10 días” (p. 2045).

Y Maeda et al. (2018) también afirma “en nuestro experimento, el entrenamiento se realizó en una PC con el sistema operativo Ubuntu 16.04 con una GPU *NVIDIA GRID K520* y 15 GB de memoria RAM”(p. 1135).

Así también hay otros como Rodríguez (2020) que realizan la comparación del tiempo de ejecución de una GPU que procesa una imagen en 0.0252 segundos y un *CPU* procesa 10 o 11 imágenes en 1 segundo. Al llevar esto a grandes cantidades de imágenes, claramente el ganador es un GPU (p. 113).

Del conjunto de datos, normalmente el entrenamiento de la red se realiza sobre el 80% del total de datos llamado train set o grupo de entrenamiento, mientras que el 20% restante se usa como set test (grupo de prueba). Se debe tener en cuenta que estos dos conjuntos de datos contienen muestras diferentes para evitar el error de *overfitting* o sobre entrenamiento.

Los errores de overfitting ocurre cuando el modelo se ha ajustado demasiado a los datos de entrenamiento, impidiendo su capacidad de generalización a nuevos casos. Esto puede deberse a un entrenamiento excesivo, una arquitectura muy compleja o una insuficiente cantidad de datos de entrenamiento. El problema de *underfitting* o sub-entrenamiento es que el modelo no ha aprendido bien los datos de entrenamiento, por lo que clasifica mal tanto los datos de entrenamiento como los de prueba Montalvo (2021).

Una vez que el conjunto de datos está listo y validado en el formato requerido de la arquitectura de red con aprendizaje supervisado, se comienza el proceso de entrenamiento.

El entrenamiento se ejecuta por un número fijo de *epochs* o épocas, donde la red ve el conjunto de datos. También se establece un batch, que es el número de instancias evaluadas antes de actualizar los pesos. Estos parámetros se eligen experimentalmente.

Después de entrenar la red neuronal en el conjunto de datos completo, evaluamos su rendimiento en el mismo conjunto. Si los resultados no son satisfactorios, ajustamos los parámetros para comprender mejor los datos y así lograr las predicciones que tenemos como objetivo. Muchos conceptos de este acápite se verán de manera practica en el capítulo III.

2.2.5 Métodos de detección de estados de Superficie de pavimento

En este acápite, revisaremos los métodos utilizados para la detección automática de los estados de la superficie de los pavimentos enfocados en daños, como grietas y/o fisuras (longitudinales, transversales y piel de cocodrilo).

El objetivo es desarrollar un modelo práctico que detecte y clasifique con precisión los diferentes tipos de daños en la superficie de los pavimentos. Para ello primero presentaremos los trabajos tradicionales sobre detección de grietas. Luego, la detección de grietas basadas en el Aprendizaje Profundo en la que se discuten los métodos para demostrar su superioridad frente a los enfoques tradicionales.

2.2.5.1 Métodos tradicionales de detección de grietas

En este trabajo de investigación, cuando nos referimos a métodos tradicionales de detección de grietas estas se basan en técnicas de aprendizaje no profundo. Numerosos artículos de investigación se han dedicado a la publicación de automatizar la detección de grietas en las superficies del pavimento. Estos trabajos se dividen en cinco categorías,

según F. Yang et al. (2019) son: “Wavelet transform” (transformación Wavelet) que en este trabajo de investigación lo vamos a generalizar con el nombre de técnicas de descomposición así como lo menciona Ríos et al. (2020), “image thresholding”(umbral de imagen), “hand crafted feature and classification”(características y clasificación manual), “edge detection based methods”(Método basado en la detección de bordes) y “minimal path based methods”(Métodos basados en rutas mínimas).

2.2.5.1.1 Técnicas de descomposición

El análisis mediante la descomposición de la señal en componentes más representativos es común en ingeniería. La idea ha progresado y las investigaciones más recientes se presentan a continuación:

En el informe realizado por Javidi et al. (2003), los autores proponen emplear la detección de bordes mediante la Transformada *Wavelet Multi-scale* y la Transformada de *Hough* para analizar imágenes de carreteras con grietas. La Transformada Wavelet permite diferenciar entre las grietas reales y las falsas detecciones ocasionadas por el ruido en el fondo de la imagen. Por su parte, la Transformada de Hough resulta útil para clasificar las grietas y estimar su número y longitud. Posteriormente, se logró utilizar una Transformada *Wavelet* 2D multiescalar para construir un mapa de coeficientes complejos y ubicar las regiones con grietas, buscando los coeficientes Wavelet más elevados, desde la escala más grande hasta la más pequeña, tal como lo proponen Zhou et al. (2005).

En el trabajo de Ouyang (2013) muestra que aplicar la transformada de *Haar* a imágenes de pavimento recopiladas con un escáner láser 3D permite detectar mejor los bordes de grietas, mejorando así la detección de grietas. Por otro lado, el trabajo de Gui et al. (2018) propone un modelo que descompone los perfiles de pavimento 3D en diferentes componentes. Esto permite caracterizar la información sobre ubicación y profundidad de señales dispersas como grietas, marcas viales, surcos y baches en perfiles.

Sin embargo, estos enfoques presentan limitaciones para tratar con grietas que poseen propiedades de baja continuidad o alta curvatura. Esto se debe al nivel de la característica anisotrópica de Wavelet, según lo expuesto por Yang et al. (2019).

2.2.5.1.2 Métodos de Umbral de Imagen

Los métodos de umbral establecen rangos para separar rasgos de interés, pero son sensibles al ruido. Se han propuesto alternativas para mejorar su desempeño.

Por ejemplo, en Chambon y Moliard (2011), los algoritmos de preprocesamiento se utilizan primero para reducir los artefactos de iluminación; luego se le aplica un umbral a la imagen para producir candidatos a grietas. En Huang et al. (2014) se establece el valor umbral en función de las condiciones de la superficie del pavimento. Con los umbrales apropiados, el método propuesto aprovecha las ventajas de las imágenes en escala de grises 2D y los métodos de datos de escaneo láser 3D y mejora su rendimiento de detección respectivo.

La imagen de grieta procesada se refina aún más utilizando tecnologías morfológicas. Por ejemplo, en Fernandes y Ciobanu (2014) presentan una variante de este grupo, en la que aprovechan los métodos basados en gráficos para el refinamiento de los candidatos de grietas.

En general, su simplicidad es una ventaja, pero son sensibles al ruido y dependen mucho de la iluminación. Por lo tanto, la desventaja de estos métodos es encontrar el umbral adecuado para separar los rasgos de interés, lo cual es especialmente difícil en imágenes de grietas con diversas condiciones ambientales y de iluminación Ríos et al. (2020).

2.2.5.1.3 Características y clasificación manual

Este método de detección de grietas tiene su particularidad de basarse en características hechas manualmente y que su clasificación sea mediante parches. En el caso que presenta Hu y Zhao (2010) propone un operador “local binary patterns (LBP)” (patrón binario local) para la detección de grietas en el pavimento en la que especifica que se clasifique en áreas uniformes y áreas rugosas. así mismo, Quintana et al. (2016) extrae las características de textura con LBP pero añade un segundo procedimiento que es “Hough transform features (HTF)” (características de transformación de hough) y que funciona en paralelo. En la etapa de fusión, la detección se completa mezclando esas técnicas y se extraen los parches altamente dañados en la carretera como descriptor de las fisuras.

La desventaja de LBP es que no proporciona resultados satisfactorios cuando el fondo tiene un patrón de textura con frecuencias altas.

2.2.5.1.4 Método basado en la detección de bordes

Dado que los bordes visualmente sobresalientes corresponden a una variedad de fenómenos visuales, es difícil encontrar un enfoque unificado para la detección de bordes. Motivados por esta observación, varios artículos de investigación han explorado diferentes técnicas para detección de bordes.

En Maode et al. (2007) emplean filtros morfológicos en la detección de grietas en el pavimento y construyen un algoritmo de filtro mediano con estructura múltiple, para evitar ruidos. así mismo, Ayenu-Prah y Attoh-Okine (2008) filtran varias imágenes con un algoritmo de descomposición de modo empírico bidimensional para eliminar el ruido y la imagen residual se analiza con el detector de bordes *Sobel* para detectar grietas.

Y más adelante, Dollár y Zitnick (2013) aplica un enfoque de aprendizaje estructurado para la detección de bordes. Y Shi et al. (2016) aplica este mismo enfoque de explotar la información de manera estructural para detectar grietas en el pavimento. Estas aplicaciones requieren eficiencia computacional.

Aunque después Yan y Yuan (2018) para mejorar la estabilidad del sistema, aplica un flujo de trabajo de procesamiento automático de datos al definir primero una “Region of interest (RoI)” (Región de Interés). El proceso incluye ejecutar primero el detector de bordes *Canny* y la transformada probabilística de *Hough* para localizar bordes. Presenta como ventaja su simplicidad y bajo costo computacional.

Estos enfoques analizan un parche de imagen para calcular la probabilidad de que el píxel central tenga un borde. Sin embargo, son sensibles al ruido y a ciertas orientaciones, dependiendo de qué operador sea utilizado Ríos et al. (2020).

2.2.5.1.5 Métodos basados en rutas mínimas

El algoritmo Snake o contorno activo de Snake introducido por Kass et al. (1988), emplea el método de ruta mínima basado en principios de minimización de energía variacional para extraer curvas abiertas simples en imágenes, definiendo dos términos clave: una energía externa, vinculada a características deseadas de la imagen (por ejemplo, bordes) para guiar la evolución de la curva, y una energía interna para mantener la suavidad de la curva durante su evolución. Kaul et al. (2012) proponen un enfoque que detecta estructuras de imagen similares a contornos utilizando una versión optimizada del método de ruta mínima que requiere menos información previa sobre la topología y los puntos finales deseados de las curvas. El algoritmo no necesita más que el conocimiento de un punto arbitrario, lo que resulta en resultados comparables con los métodos de ruta mínima anteriores. Por otro lado, Amhaz et al. (2016) presentan un método de detección de grietas que emplea un algoritmo de selección de ruta mínima en dos etapas: primero, identifica localmente los píxeles significativos como puntos finales y calcula las rutas mínimas entre estos puntos, y luego selecciona las rutas de menor costo a escala global.

Los contornos activos separan bien las fallas, son resistentes al ruido y a bordes falsos, pero requieren una inicialización cercana al objeto y tienen problemas con las concavidades Ríos et al. (2020).

2.2.5.2 Métodos de detección de grietas basadas en aprendizaje profundo

El aprendizaje profundo utiliza redes neuronales que pueden extraer y clasificar características automáticamente, a diferencia de los métodos tradicionales de inteligencia computacional donde las características deben definirse manualmente Arya et al. (2021).

En los últimos años, el aprendizaje profundo ha logrado resultados impresionantes en visión artificial. Sus redes neuronales profundas han demostrado ser muy efectivas para procesar millones de imágenes de alta resolución de manera rápida y precisa. Esto ha permitido avances significativos en diversos campos de investigación. La red neuronal convolucional (CNN) es una de sus principales arquitecturas que tiene la ventaja que después de cada entrenamiento, detecta automáticamente las características críticas sin supervisión humana (Krizhevsky , 2017).

Es importante tener en cuenta que la clasificación se refiere a etiquetar una imagen en lugar de un objeto, mientras que la detección significa asignar una etiqueta a una imagen e identificar las coordenadas del objeto, como lo demuestra la competencia *ImageNet* (Deng et al., 2009).

Por esta razón, ha surgido un gran interés entre los investigadores en aplicar el aprendizaje profundo y sus principales arquitecturas en la tarea de detección de grietas en el pavimento. En el estudio de Zhang et al. (2016) primero propusieron una red neuronal relativamente poco profunda que consta de cuatro capas convolucionales y dos capas completamente conectadas, para realizar la detección de grietas en una forma basada en parches de imagen anotados manualmente y adquiridos por un sensor de bajo costo, es decir, un teléfono inteligente. Además, comparan este enfoque con métodos tradicionales basados en características diseñadas manualmente para resaltar la superioridad del aprendizaje profundo en la detección precisa de grietas en pavimentos. Paulyet al. (2017) aplican una red neuronal más profunda para clasificar los parches con fisuras y sin fisuras y demuestran la efectividad del uso de las redes neuronales más profundas para mejorar la precisión.

Feng et al. (2017) proponen un sistema de aprendizaje activo profundo para hacer frente al problema de recursos limitados de etiquetas. En primer lugar, se diseña una red residual

profunda para la detección y clasificación de grietas en una imagen. Siguiendo la estrategia de aprendizaje activo, esta red se entrena tan pronto como esté disponible un conjunto inicial de imágenes etiquetadas y con ello se logra maximizar el rendimiento. En el mismo año Eisenbach et al. (2017) proporciona un conjunto de datos de deterioro del pavimento disponible gratuitamente de un tamaño lo suficientemente grande como para entrenar redes de aprendizaje profundo y primero evalúan y analizan enfoques de vanguardia en la detección de fallas en el pavimento. Además, analizo la efectividad de las técnicas de regularización de última generación obteniendo los mejores resultados utilizando solo abandonos, seguidos de solo normalización por lotes.

Los enfoques antes mencionados tratan la detección de grietas como una tarea de clasificación basada en parches. Cada imagen se recorta en pequeños parches, luego se entrena una red neuronal profunda para clasificar cada parche como *crack* o no. Esta forma es inconveniente y sensible a la escala del parche. A tal efecto se desarrolló rápido la tarea de segmentación semántica con Long et al. (2017), Badrinarayanan et al. (2017), Fan et al. (2018) y Schmugge et al. (2017) que presentan un método de segmentación de grietas en *SegNet* Badrinarayanan et al. (2015) para videos de examen visual remoto. Este método realiza la detección de grietas agregando la probabilidad de grietas de múltiples cuadros superpuestos en un video.

Recientemente Yang et al. (2022) propuso un innovador método de anotación denso y redundante para detectar las características estructurales de las grietas del pavimento de asfalto y las grietas selladas. En la que se utilizó un método semiautomático de anotación de grietas y grietas selladas para mejorar la eficacia de la creación del conjunto de datos.

2.2.6 Métricas de evaluación de la detección con aprendizaje profundo

Como se ha generado un modelo de clasificación con alguno de los algoritmos detallados en las secciones anteriores u otros diferentes. Será importante conocer ¿cuál modelo de aprendizaje profundo tiene el mejor rendimiento?, por ello es necesario utilizar herramientas que nos permitan comparar diferentes algoritmos, incluso si son de naturaleza distinta.

Por consiguiente, es de suma importancia en la presente investigación evaluar el rendimiento del algoritmo de detección de objetos a través de la matriz de confusión, lo que permite determinar si nuestro modelo es aceptable para el problema planteado.

Una matriz de confusión muestra cuántos patrones se clasificaron correcta e incorrectamente, y las etiquetas asignadas a cada uno. En su narrativa, Rodríguez (2020) expone un caso simplificado que consta de dos clases (clase A y clase B) en que nuestro modelo etiqueta cada uno de los patrones.

Los elementos que componen la matriz de confusión en este caso son:

- Verdaderos positivos (VP): número de patrones que pertenecen a la clase A y fueron correctamente clasificados como tales.
- Falsos positivos (FP): número de patrones que no pertenecen a la clase A, pero que fueron clasificados incorrectamente como pertenecientes a la clase A.
- Verdaderos negativos (VN): número de patrones que no pertenecen a la clase A y que se clasificaron correctamente como tales.
- Falsos negativos (FN): número de patrones que pertenecen a la clase A, pero que se clasificaron incorrectamente como no pertenecientes a la clase A.

Tras la elaboración de la matriz de confusión, se emplean diversas métricas de evaluación probabilísticas. Estas comparan las probabilidades predichas por el modelo con las etiquetas reales. Esto permite observar las áreas en las que el modelo de aprendizaje profundo manifiesta alta o baja confianza en sus predicciones, e identificar las clases que son confundidas con mayor frecuencia.

De la comparación entre las probabilidades predichas y las etiquetas reales, se calculan las principales métricas de evaluación utilizadas en el aprendizaje profundo, como detalla Galdi y Tagliaferri (2019). En este caso, se profundizará en aquella métrica de interés para esta tesis.

Uno de los objetivos en primera instancia de este estudio es proporcionar una medida general del rendimiento del modelo de aprendizaje profundo. A continuación, se presenta dicha métrica:

2.2.6.1 Precisión o *accuracy*

La precisión (accuracy), es una métrica que se utiliza para evaluar el rendimiento global de un modelo de aprendizaje profundo. En el contexto de la matriz de confusión, se refiere a la proporción de predicciones correctas (verdaderos positivos y verdaderos negativos) en relación con el total de predicciones realizadas. En términos Matemáticos, se representa en la Ecuación 15.

$$Precisión\ (accuracy)\ = \frac{número\ de\ predicciones\ correctas}{número\ total\ de\ predicciones} \qquad (15)$$

La métrica mencionada se limita a proporcionar una visión general del rendimiento del modelo, sin diferenciar entre las clases o el contexto en el que se evalúa. Por cual, se complementa dicho estudio con métricas que proporcionan una evaluación más detallada y equilibrada del rendimiento del modelo en cada clase y en diferentes contextos.

2.2.6.2 Precisión (Por clase)

La precisión por clase evalúa qué tan bien el modelo de aprendizaje profundo identifica correctamente los casos positivos, minimizando los falsos positivos. Utilizando los elementos de la matriz de confusión, la métrica se define como el número de verdaderos positivos dividido por la suma de verdaderos positivos y falsos positivos.

En la Ecuación 16, se observa claramente que se trata de una métrica que mide la proporción de predicciones positivas que fueron correctas.

$$Precisión_{clase} = \frac{VP}{VP + FP} \qquad (16)$$

2.2.6.3 Recall

Recall es una métrica que evalúa si el modelo de aprendizaje profundo etiqueta correctamente los casos positivos de cada clase específica, minimizando los falsos negativos. Utilizando los elementos de la matriz de confusión, la métrica se define como el número de verdaderos positivos (TP) sobre el total de positivos reales (verdaderos positivos más falsos negativos).

En la Ecuación 17, queda claro que es una métrica que mide cuántos de los casos positivos reales fueron correctamente etiquetados.

$$Recall\ = \frac{VP}{VP + FN} \qquad (17)$$

2.2.6.4 F - Score

El F-Score es una métrica que evalúa el rendimiento de un modelo de aprendizaje profundo combinando la precisión por clase y el recall. En este estudio buscamos un equilibrio entre la precisión por clase y el recall, esto lo resalto porque existe una generalización del F-Score llamada F-Beta Score, donde se introduce un parámetro β que en nuestro caso será igual 1. Esto permite igualar la importancia de precisión por clase y recall, lo que resulta en el conocido como F1-Score.

En la Ecuación 18, presentamos la fórmula general para el F-Beta Score.

$$F_\beta = (1+\beta^2) \times \frac{Precisión_{clase} \times Recall}{(\beta^2 \times Precisión_{clase}) + Recall} \quad (18)$$

En la Ecuación 19, tomamos β=1. La fórmula se convierte en:

$$F_\beta = (1+1^2) \times \frac{Precisión_{clase} \times Recall}{(1^2 \times Precisión_{clase}) + Recall} \quad (19)$$

Finalmente, en la Ecuación 20, simplificando y sustituyendo los términos de precisión y recall, obtenemos F1-Score.

$$F_1 = \frac{2VP}{2VP + FP + FN} \quad (20)$$

2.3 Definición de términos

La teoría científica es un conjunto de afirmaciones lógicamente relacionadas y respaldadas empíricamente, junto con los supuestos del investigador sobre su método y datos (Reidl-Martínez, 2012).

Por ello es fundamental el desarrollo del marco conceptual, en la que se abordara perspectivas o enfoques teóricos empleados en estudios relacionados con la presente investigación. Describimos a continuación la terminología básica a utilizar en la presente tesis:

- Aprendizaje profundo: es un tipo de aprendizaje automático que utiliza redes neuronales artificiales con múltiples capas para aprender automáticamente patrones a partir de datos que le permiten realizar predicciones o clasificaciones precisas en varios dominios, como el reconocimiento de imágenes, lo que propicia la detección automática de objetos con un alto grado de precisión.

- Condición de las superficies de pavimento flexible: se refiere a su calidad general, incluida la salud estructural, textura, deformaciones, y daños como grietas o baches en la superficie del pavimento flexible. Es crucial tener una condición óptima para asegurar la seguridad vial, y por ello que se recomienda mantenimientos periódicos, que van desde inspecciones visuales hasta tecnologías avanzadas, para evaluar y gestionar eficazmente el estado de la superficie del pavimento.

- Software de detección autónoma: son programas de cómputo desarrollados con inteligencia artificial que permiten detectar patrones de forma en imágenes y videos

sin intervención humana. Para el caso de pavimentos, hay varias opciones de software disponibles para la detección de grietas, por ejemplo: *Crackify*, *RoadDoctor*, y Pavimenta2.

- Precisión en la detección de fallas en pavimentos: los sistemas de detección de grietas en el pavimento, en particular aquellos que utilizan algoritmos avanzados como el aprendizaje profundo, lograran alcanzar altas tasas de precisión, alcanzando el 92.76 % en entornos controlados. Sin embargo, las condiciones del mundo real plantean desafíos, lo que podría reducir la precisión debido a factores como la iluminación y las distorsiones de la imagen. Los avances tecnológicos continuos están mejorando la confiabilidad, pero lograr una precisión del 100% sigue siendo un desafío. A pesar de esto, los sistemas modernos mejoran significativamente las prácticas de mantenimiento del pavimento, siendo esenciales la validación y las pruebas periódicas para garantizar la eficacia.

CAPÍTULO III: DESARROLLO DEL TRABAJO DE LA TESIS

Como vimos en el Capítulo II, el daño al pavimento es el principal factor que afecta el desempeño de la carretera. El agrietamiento del pavimento es uno de los factores que exigen un constante mantenimiento de carreteras. Con el fin de lograr una clasificación precisa de grietas, segmentación y parámetros geométricos cálculo, en este capítulo proponemos la metodología de creación de algoritmos de aprendizaje profundo con el uso de redes neuronales convolucionales para detectar grietas en pavimentos. Estos algoritmos serán entrenados en una base de datos original, y que hemos obtenido por cuenta propia en la ciudad de Piura.

Como primera etapa estableceremos las categorías de detección, según las cuales entrenaremos a nuestros algoritmos para que detecten y clasifiquen grietas con una alta confianza, en términos probabilísticos. En una siguiente etapa, pasaremos a definir los algoritmos, sus parámetros y la precisión deseada. Es importante también seleccionar la porción de datos que serán utilizados para el entrenamiento y para la prueba, este proceso influye directamente en la segmentación que el modelo llevará a cabo.

Finalmente, de los cuatro modelos desarrollados específicamente para esta tesis, el progreso gradual en la precisión de detección fue un elemento clave para demostrar el enfoque investigativo desarrollado en este trabajo. Se observará que los algoritmos de aprendizaje profundo empleados en este estudio logran detectar grietas transversales, longitudinales y de piel de cocodrilo con una precisión superior al 90%, destacando un método óptimo que alcanzó una precisión del 92.76%, el cual fue utilizado en esta investigación.

3.1 Análisis de los Datos y Resultados

En esta tesis, el enfoque se concentra únicamente en la detección de fallas longitudinales, transversales y de tipo piel de cocodrilo. Este último es un tipo de falla muy común en los pavimentos a nivel nacional, como se mencionó en la sección 2.2.2.1. En este apartado, se enfatiza que cada tipo de falla posee patrones clave que enseñan al algoritmo de aprendizaje profundo para su detección. Estas serían las características visuales específicas de cada falla, las cuales se muestran en las Figuras 40, 41 y 42.

En el anexo 4 se muestra detalladamente el proceso por el cual pasaron las imágenes, que finalmente se utilizaron en el entrenamiento del modelo de aprendizaje profundo. Dicho proceso permitió preparar adecuadamente el conjunto de datos de entrenamiento.

Figura 40

Falla de Tipo Longitudinal

Figura 41

Falla tipo transversal

Figura 42

Falla tipo piel de cocodrilo

3.1.1 Procesamiento de la data

3.1.1.1 Obtención de la data

La data ha sido recopilada en pavimentos de la ciudad de Piura, como se observa en el Anexo 5. Hemos dividido nuestros conjuntos de datos en 2 categorías (directorios):

- **train:** que contiene las fotos a ser usadas en la etapa de entrenamiento de la red neuronal, grupo de entrenamiento ≈ 80%.
- **test:** que contiene las fotos a ser usadas en la etapa de prueba de la precisión del algoritmo, grupo de prueba ≈ 20%.

En las figuras siguientes mostramos la división de data entre train y test, para nuestras tres categorías:

Figura 43

Repartición de data para la categoría: Fallas Longitudinales

```
! ls /content/drive/MyDrive/data_piura/test/longitudinal/ | wc -l
```

```
221
```

```
! ls /content/drive/MyDrive/data_piura/train/longitudinal/ | wc -l
```

```
883
```

Figura 44

Repartición de data para la categoría: Fallas Transversales

```
! ls /content/drive/MyDrive/data_piura/test/transversal/ | wc -l
```

```
221
```

```
! ls /content/drive/MyDrive/data_piura/train/transversal/ | wc -l
```

```
883
```

Figura 45

Repartición de data para la categoría: Fallas tipo piel de cocodrilo

```
! ls /content/drive/MyDrive/data_piura/test/cocodrilo/ | wc -l
```

```
221
```

```
! ls /content/drive/MyDrive/data_piura/train/cocodrilo/ | wc -l
```

```
883
```

3.1.1.2 Creación de los modelos

En la presente tesis usaremos la librería imageAI para crear nuestros modelos de aprendizaje profundo con la ayuda del aprendizaje transferido. Para el desarrollo de nuestros modelos usaremos Python, usando librerías que podemos instalar:

Figura 46

Librerías a utilizar para crear modelos de aprendizaje profundo

```
import os
from imageai.Classification import ImageClassification
from imageai.Detection import ObjectDetection
```

Para instalar la librería **imageAI** utilizar el siguiente comando en un ambiente Python:

```
pip3 install imageai ––upgrade
```

3.1.1.3 Proceso de entrenamiento

Una vez instalada la librería en nuestro sistema, debemos elegir cómo realizar el entrenamiento, esto conlleva dos pasos: elegir el equipo computacional a utilizar, y el modelo a partir del cual transferir el conocimiento a nuestra nueva red neuronal.

3.1.1.3.1 Equipos utilizados.

En esta tesis hemos utilizados tarjetas gráficas para poder procesar la gran cantidad de datos de fotografías, hemos alquilado los siguientes dos equipos:

- Tarjeta gráfica *NVIDIA Tesla T4*, la cual tiene 16GB de memoria RAM, el chip es de tecnología TSMC de 12 nm. Esta tarjeta se basa en la arquitectura *NVIDIA Turing*™ y tiene una conexión *PCIe* de 70 vatios de bajo consumo, *T4* está optimizado para entornos informáticos convencionales y cuenta con *Turing Tensor Cores* de precisión múltiple y nuevos *RT Cores*. Combinado con pilas de software en contenedores acelerados de *NGC*, T4 ofrece un rendimiento revolucionario a escala, tal como se muestra en la Figura 47.

Figura 47

Tarjeta gráfica de NVIDIA Tesla T4 con 16GB de RAM

Nota. Fuente https://is.gd/01Zrev (Página web de NVIDIA con acceso abierto de Productos disponibles).

- Tarjeta gráfica *AMD modelo RTX 6900 XT*, la cual tiene 16GB de memoria RAM, el chip es de tecnología *TSMC* de 7 nm. Esta tarjeta se compone de un chip grande con un área de matriz de 520 mm² y 26.800 millones de transistores. Cuenta con 5120 unidades de sombreado y 320 unidades de mapeo de texturas. La tarjeta también tiene 80 núcleos de aceleración de trazado de rayos. GPU funciona a una

frecuencia de 1825 MHz, que logra aumentar hasta 2250 MHz, la memoria funciona a 2000 MHz (16 Gbps efectivos), tal como se muestra en la Figura 48.

Figura 48

Tarjeta gráfica de AMD RTX 6900XT con 16GB de RAM

Nota. Fuente https://is.gd/zvreWY (Página web de AMD con acceso abierto de Productos disponibles).

3.1.1.3.2 Creación del modelo de aprendizaje profundo

Siguiendo el marco teórico que desarrollamos en el capítulo 2, seleccionamos la red neuronal DenseNet como modelo base para implementar la transferencia de aprendizaje y desarrollar un nuevo modelo adaptado a nuestra data. Elegimos este modelo debido a las siguientes razones:

- Esta fue entrenada prioritariamente con fotografías tomadas con móvil, así como en nuestro caso.
- De otros modelos que utilizamos como base, este fue el que mejor precisión nos brindó.

Para el entrenamiento utilizamos el código que se muestra en la Figura 49:

Figura 49

Código para generar un modelo mediante transferencia de aprendizaje

```
model_trainer = ClassificationModelTrainer()
model_trainer.setModelTypeAsResNet50()
model_trainer.setDataDirectory(r"/content/drive/MyDrive/data_piura/")
model_trainer.trainModel(num_experiments=100, batch_size=32)
```

En el código de la Figura 49, lo que hemos hecho es generar 100 épocas de entrenamiento, cada una con 32 sub-etapas de refinamiento. El resultado del entrenamiento se muestra en la Figura 50, donde mostramos los resultados de las últimas 3 épocas. Vemos que nuestro modelo obtiene una precisión de 92.76%.

Figura 50

En las últimas 3 épocas de entrenamiento, la mejor precisión fue 92.76%

```
----------
100%|██████████| 56/56 [00:14<00:00,  3.90it/s]
train Loss: 0.0747 Accuracy: 0.9711
100%|██████████| 14/14 [00:02<00:00,  4.77it/s]
test Loss: 0.2493 Accuracy: 0.9027
Epoch 98/100
----------
100%|██████████| 56/56 [00:14<00:00,  3.91it/s]
train Loss: 0.1081 Accuracy: 0.9564
100%|██████████| 14/14 [00:02<00:00,  4.83it/s]
test Loss: 0.2746 Accuracy: 0.8959
Epoch 99/100
----------
100%|██████████| 56/56 [00:14<00:00,  3.85it/s]
train Loss: 0.0911 Accuracy: 0.9677
100%|██████████| 14/14 [00:03<00:00,  3.75it/s]
test Loss: 0.2575 Accuracy: 0.8801
Epoch 100/100
----------
100%|██████████| 56/56 [00:14<00:00,  3.89it/s]
train Loss: 0.1035 Accuracy: 0.9666
100%|██████████| 14/14 [00:02<00:00,  4.78it/s]test Loss: 0.2534 Accuracy: 0.8824
Training completed in 29m 16s
Best test accuracy: 0.9276
```

En la Figura 50, se muestra cómo se computó la métrica de precisión (accuracy) utilizando el grupo de prueba en cada una de las épocas del entrenamiento. A continuación, se presenta un ejemplo que explica el procedimiento utilizado para el cálculo de esta métrica. En este caso, se selecciona la época de entrenamiento correspondiente a la mejor precisión de prueba (best test accuracy), en la cual se aplicó la Ecuación 15:

$$Precisión = \frac{\text{total de predicciones correctas}}{\text{total de imágenes en el grupo de prueba}} = \frac{615}{663} = 0.9276 = 92.76\ \%$$

En el Anexo 3, se presenta el cálculo de las iteraciones que se llevan a cabo, época por época, hasta que el modelo alcanza la convergencia hacia el mejor resultado de precisión. En este caso, se detallan las 100 épocas de entrenamiento necesarias para lograr este objetivo.

3.1.2 Convergencia del modelo

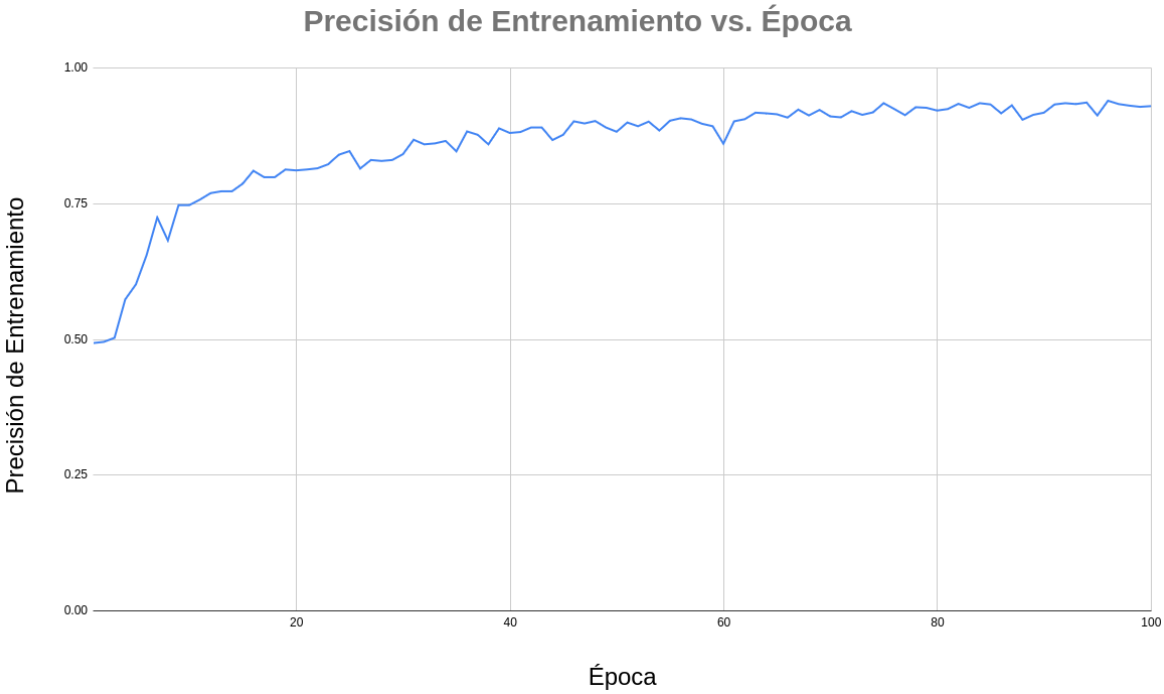
De los resultados de entrenamiento del mejor modelo, vemos que convergió a una precisión de 92.76%, este modelo lo podemos exportar como formato .h5, el cual luego podemos cargar en un ambiente Python y utilizarlo para hacer predicciones en fotografías que el modelo no haya visto aún.

En la Figura 51 graficamos la evolución de la precisión del algoritmo a lo largo de las 100 épocas de entrenamiento. En esta gráfica, se observa cómo la precisión del modelo fue aumentando gradualmente con cada época de entrenamiento hasta converger aproximadamente al 93%, lo cual se considera un nivel de precisión muy alto.

Figura 51

En la época 100, la evolución de la precisión alcanzo alrededor del 93%

3.1.3 Analizando el algoritmo en imágenes nuevas



Una vez creado el modelo final en formato .h5, le ponemos un nombre, en este caso:

`densenet_hilder,`

Y de manera similar los otros archivos de configuración que exporta imageAI. Luego podemos probarlo en Python usando imágenes de nuestra carpeta reservada para pruebas adicionales. Para ello usaremos las siguientes funciones de ImageAI:

- `setModelTypeAsDenseNet121`: para poner el modelo base que usamos para extender con aprendizaje transferido.
- `classifyImage`: esta función se aplica a una nueva fotografía y retorna las probabilidades que el algoritmo asigna a cada categoría, la que tiene la mayor probabilidad será la que el algoritmo considere que observa en la imagen.

Los resultados obtenidos del algoritmo de detección de fallas en pavimentos flexibles se muestran en las Figuras 52, 53 y 54, cada una correspondiente a una categoría de falla analizada. Es importante destacar que el código utilizado es de libre acceso y está disponible para ser probado con cualquier otra imagen tomada de un pavimento flexible.

Figura 52

El algoritmo detectó una falla longitudinal con un 89% de precisión

```
from imageai.Classification.Custom import CustomImageClassification

prediction = CustomImageClassification()
base_dir="/content/drive/MyDrive/data_piura/"
trained_model=base_dir+"models/densenet_hilder/"

json_model=base_dir+"models/densenet_hilder/data_piura_model_classes.json"
execution_path = os.getcwd()

prediction.setModelTypeAsDenseNet121()

prediction.setModelPath(trained_model)
prediction.setJsonPath(json_model)
prediction.loadModel()

predictions, probabilities = prediction.classifyImage(os.path.join(execution_path,
                  base_dir+"test/longitudinal/imagenes_prueba_fallas_longitudinal_0.jpg"),
                    result_count=2)

for eachPrediction, eachProbability in zip(predictions, probabilities):
    print(eachPrediction , ": " , eachProbability)
```

```
longitudinal: 0.89
transversal: 0.63
cocodrilo: 0.29
```

Figura 53

El algoritmo detectó una falla Transversal con un 78% de precisión

```
from imageai.Classification.Custom import CustomImageClassification

prediction = CustomImageClassification()
base_dir="/content/drive/MyDrive/data_piura/"
trained_model=base_dir+"models/densenet_hilder/"

json_model=base_dir+"models/densenet_hilder/data_piura_model_classes.json"
execution_path = os.getcwd()

prediction.setModelTypeAsDenseNet121()

prediction.setModelPath(trained_model)
prediction.setJsonPath(json_model)
prediction.loadModel()

predictions, probabilities = prediction.classifyImage(os.path.join(execution_path,
                  base_dir+"test/cocodrilo/imagenes_prueba_fallas_cocodrilo_0.jpg"),
                    result_count=2)

for eachPrediction, eachProbability in zip(predictions, probabilities):
    print(eachPrediction , ": " , eachProbability)
```

```
transversal: 0.78
longitudinal: 0.55
cocodrilo: 0.48
```

Figura 54

El algoritmo detectó una falla de piel de cocodrilo con un 87% de precisión

```
from imageai.Classification.Custom import CustomImageClassification

prediction = CustomImageClassification()
base_dir="/content/drive/MyDrive/data_piura/"
trained_model=base_dir+"models/densenet_hilder/"

json_model=base_dir+"models/densenet_hilder/data_piura_model_classes.json"
execution_path = os.getcwd()

prediction.setModelTypeAsDenseNet121()

prediction.setModelPath(trained_model)
prediction.setJsonPath(json_model)
prediction.loadModel()

predictions, probabilities = prediction.classifyImage(os.path.join(execution_path,
                  base_dir+"test/cocodrilo/imagenes_prueba_fallas_cocodrilo_0.jpg"),
                    result_count=2)

for eachPrediction, eachProbability in zip(predictions, probabilities):
    print(eachPrediction , ": " , eachProbability)
```

```
cocodrilo: 0.87
longitudinal: 0.54
transversal: 0.37
```

3.2 Discusión e Interpretación de Resultados

Vemos que el algoritmo no se equivoca en ninguno de los casos al ser probado con imágenes con las que no ha sido entrenado, obteniendo los siguientes resultados:

- Detección de falla de tipo longitudinal con precisión de 89%.
- Detección de falla de tipo transversal con precisión de 78%.
- Detección de falla tipo piel de cocodrilo con precisión de 87%.

Utilizando la funcionalidad:

```
imageai.Detection,
```

con el código proporcionado en las Figuras 55, 57 y 59 para cada categoría de falla podemos exportar imágenes en las cuales al ingresar una fotografía de entrada obtenemos una fotografía de salida se obtiene una fotografía de salida que contiene un rectángulo que indica el tipo de falla detectada y la probabilidad asociada que el algoritmo le otorgó a dicha etiqueta, como se muestra en las Figuras 56, 58 y 60. A continuación, se muestra cómo hacerlo para cada tipo de falla:

Figura 55

Código para detectar y enmarcar falla de tipo longitudinal en una fotografía

```
from imageai.Detection.Custom import CustomObjectDetection

detector = CustomObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath("densenet_hilder.pt")
detector.setJsonPath("densenet_hilder.json")
detector.loadModel()
detections = detector.detectObjectsFromImage(input_image="prueba_longitudinal.jpg",
                                             output_image_path="prueba_longitudinal.jpg")
for detection in detections:
    print(detection["name"], " : ",
          detection["percentage_probability"], " : ", detection["box_points"])
```

Figura 56

El algoritmo detecta una falla longitudinal con una probabilidad del 68%.

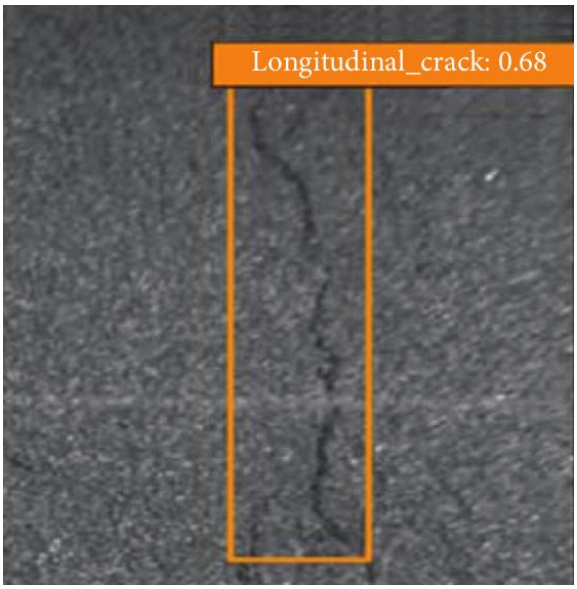


Figura 57

Código para detectar y enmarcar falla de tipo transversal en una fotografía

```
from imageai.Detection.Custom import CustomObjectDetection

detector = CustomObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath("densenet_hilder.pt")
detector.setJsonPath("densenet_hilder.json")
detector.loadModel()
detections = detector.detectObjectsFromImage(input_image="prueba_transversal.jpg",
                                             output_image_path="prueba_transversal.jpg")
for detection in detections:
    print(detection["name"], " : ",
          detection["percentage_probability"], " : ", detection["box_points"])
```

Figura 58

El algoritmo detecta una falla transversal con una probabilidad del 88%.

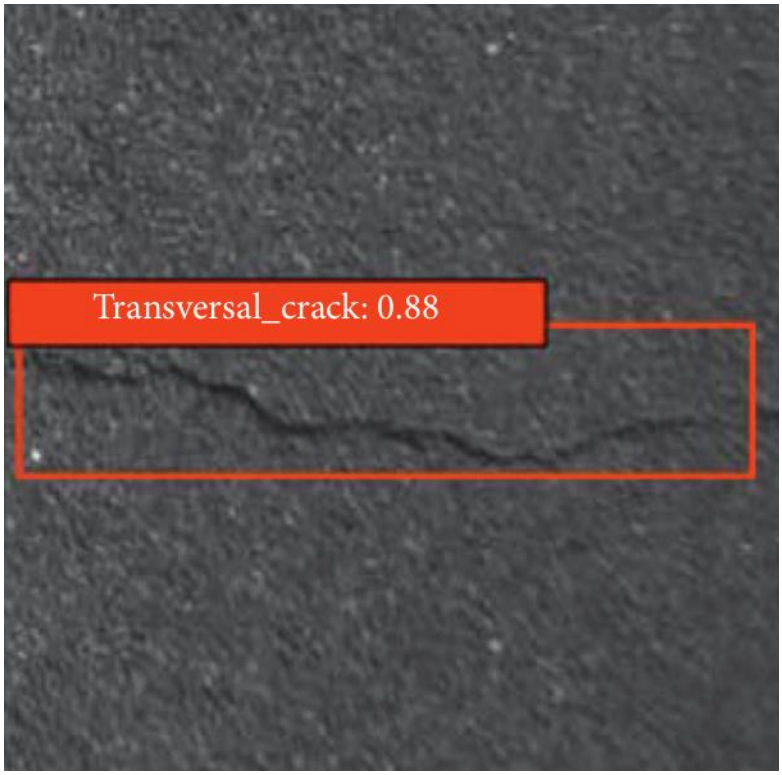


Figura 59

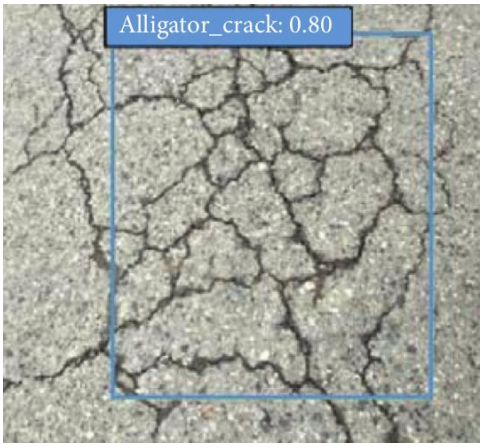
Código para detectar y enmarcar falla de piel de cocodrilo en una fotografía

```
from imageai.Detection.Custom import CustomObjectDetection

detector = CustomObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath("densenet_hilder.pt")
detector.setJsonPath("densenet_hilder.json")
detector.loadModel()
detections = detector.detectObjectsFromImage(input_image="prueba_cocodrilo.jpg",
                                             output_image_path="prueba_cocodrilo.jpg")
for detection in detections:
    print(detection["name"], " : ",
          detection["percentage_probability"], " : ", detection["box_points"])
```

Figura 60

El algoritmo detecta una falla piel de cocodrilo con una probabilidad del 80%.



3.3 Contrastación de la hipótesis

3.3.1 Validación de resultados con el software Pavimenta2.

Para validar nuestro modelo autónomo de detección de fallas en el pavimento de Piura, a continuación, presentamos un experimento donde comparamos nuestro software con Pavimenta2, un programa creado por el Banco Interamericano de Desarrollo (BID, 2022), el cual es un innovador software de inteligencia artificial diseñado para detectar, clasificar y cuantificar diversos problemas en el pavimento de las carreteras y evaluar la señalización del tráfico. Se desarrolló para permitir a entidades públicas y privadas procesar secuencias de vídeo de carreteras y generar informes detallados sobre defectos del pavimento y señalización vertical, con resultados visibles en plataformas de sistemas de información geográfica. Pavimenta2 también facilita la estimación de los costes de mantenimiento asociados a las averías identificadas.

Al automatizar la documentación, la medición y el registro de los defectos de las carreteras, Pavimenta2 reduce significativamente el tiempo de procesamiento de años a semanas, mejorando la resiliencia de las carreteras al permitir reparaciones tempranas y prevenir situaciones críticas. La integración de la inteligencia artificial en las estrategias de mantenimiento de carreteras agrega un valor sustancial y, en última instancia, ofrece oportunidades para optimizar y modernizar la gestión de carreteras en beneficio de todas las partes interesadas y de la sociedad en general.

En la Figura 61, observamos los comandos de Python que utiliza Pavimenta2 para cargar la base de datos y crear un procesador inteligente.

Figura 61

Cargando la carpeta de base de datos en el software Pavimentados

```
### Image with the GPS data embebed
route = Path("/home/pavimentados/fotos_hilder")
```

```
ml_processor = MultiImage_Processor(assign_devices = False, gpu_enabled = False)
```

```
2023-11-05 12:26:33.032299: W tensorflow/stream_executor/platform/default/dso_loade
2023-11-05 12:26:33.032459: W tensorflow/stream_executor/cuda/cuda_driver.cc:263] f
2023-11-05 12:26:33.032478: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:1
2023-11-05 12:26:33.033644: I tensorflow/core/platform/cpu_feature_guard.cc:193] Th
To enable them in other operations, rebuild TensorFlow with the appropriate compile
100%|██████████| 16/16 [00:00<00:00, 157.59it/s]
100%|██████████| 317/317 [00:00<00:00, 393.56it/s]
Createing siamese model
Loading Weights siamese
Siamese model first loaded
1/1 [==============================] - 0s 149ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 37ms/step
10/10 [==============================] - 0s 45ms/step
10/10 [==============================] - 0s 43ms/step
```

Pavimenta2 utiliza algoritmos de inteligencia artificial para detectar, clasificar y cuantificar automáticamente varios tipos de defectos del pavimento, como grietas, baches y señalización. Al analizar vídeos o imágenes de la superficie de la carretera, el software logra identificar estos defectos con gran precisión y eficiencia. En la Figura 62 se muestra como el *workflow* del programa ejecuta el procesador de inteligencia artificial, el cual empieza con su etapa de entrenamiento con la base de datos otorgada.

Figura 62

Entrenamiento en Pavimenta2

```
results = workflow.execute(ml_processor)
```

```
  0%|          | 0/16 [00:00<?, ?it/s]
1/1 [==============================] - 2s 2s/step
  6%|▌         | 1/16 [00:09<02:29,  9.95s/it]
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 113ms/step
 12%|█         | 2/16 [00:19<02:16,  9.73s/it]
1/1 [==============================] - 1s 1s/step
 19%|██        | 3/16 [00:28<02:01,  9.36s/it]
1/1 [==============================] - 1s 1s/step
 25%|██▌       | 4/16 [00:37<01:49,  9.16s/it]
1/1 [==============================] - 1s 1s/step
 31%|███       | 5/16 [00:46<01:39,  9.09s/it]
1/1 [==============================] - 1s 1s/step
 38%|███▌      | 6/16 [00:55<01:30,  9.03s/it]
1/1 [==============================] - 1s 1s/step
 44%|████      | 7/16 [01:04<01:22,  9.13s/it]
```

Uno de los beneficios clave de Pavimenta2 es que reduce el tiempo y el esfuerzo necesarios para los entrenamientos, teniendo ya una inteligencia base, que el BID proporciona de manera gratuita para que los programadores la usen para extender su funcionalidad. Además, el software genera informes completos con análisis detallados de los defectos del pavimento y la señalización vial, proporcionando información valiosa para la gestión de la infraestructura y la planificación del mantenimiento. Los usuarios poseen la posibilidad de visualizar los resultados de forma georreferenciada utilizando plataformas de sistemas de información geográfica (*GIS*), lo cual facilita el análisis espacial y la toma de decisiones. Por ejemplo, en la Figura 63 se observa la tabla resultada luego de que el software finaliza su entrenamiento.

Figura 63

Tabla de resultados - Pavimentado2

	section	Grieta Lineal Longitudinal	Intervalo Lineal Longitudinal	Grieta Lineal Transversal	Intervalo Lineal Transversal	Piel de Cocodrilo	Protuberancia, Bache	Desenfoque Paso Peatonal	Desenfoque Linea Blanca	Otras Fallas	latitude	longitude
0	0.0	0.0	0.0	0.0	0.0	155.0	0.0	0.0	0.0	0.0	-5.201314	-80.633592
1	1.0	0.0	0.0	0.0	0.0	485.0	0.0	0.0	0.0	0.0	-5.200728	-80.634867
2	2.0	0.0	0.0	0.0	0.0	554.0	0.0	0.0	0.0	0.0	-5.199153	-80.638206
3	6.0	0.0	0.0	0.0	0.0	243.0	0.0	0.0	0.0	0.0	-5.200972	-80.634336
4	14.0	0.0	0.0	0.0	0.0	101.0	0.0	0.0	0.0	0.0	-5.199694	-80.636897
5	15.0	0.0	0.0	0.0	0.0	0.0	325.0	0.0	0.0	0.0	-5.199275	-80.637953
6	24.0	0.0	0.0	0.0	0.0	485.0	0.0	0.0	0.0	0.0	-5.201469	-80.633269
7	26.0	0.0	0.0	0.0	0.0	0.0	172.0	0.0	0.0	0.0	-5.201117	-80.634033
8	39.0	0.0	0.0	0.0	0.0	39.0	0.0	0.0	0.0	0.0	-5.199861	-80.636653
9	52.0	0.0	0.0	0.0	0.0	389.0	0.0	0.0	0.0	0.0	-5.201406	-80.633403
10	67.0	0.0	0.0	0.0	0.0	749.0	0.0	0.0	0.0	0.0	-5.201419	-80.633369
11	72.0	0.0	0.0	0.0	0.0	373.0	0.0	0.0	0.0	0.0	-5.201025	-80.634217
12	74.0	0.0	0.0	0.0	0.0	775.0	0.0	0.0	0.0	0.0	-5.201389	-80.633436
13	86.0	0.0	0.0	0.0	0.0	775.0	0.0	0.0	0.0	0.0	-5.201178	-80.633897
14	91.0	0.0	0.0	0.0	0.0	0.0	700.0	0.0	0.0	0.0	-5.201131	-80.634003
15	94.0	0.0	0.0	0.0	0.0	538.0	0.0	0.0	0.0	0.0	-5.201356	-80.633503

Pavimenta2 también ofrece una interfaz capaz de darnos la foto en la cual se produce la detección de la falla, y la categoriza usando diferentes etiquetas, como se muestra en la Figura 64.

Los resultados del software Pavimentados también se exportan en formato GIS. En nuestro caso, hemos escrito un pequeño programa en Python que nos permite integrar directamente un mapa libre y poner ahí los puntos que Pavimenta2 detecta como fallas potenciales (puntos rojos). Ver la tabla en la Figura 63 como referencia de cómo se obtiene la data geográfica. En la Figura 65 vemos cómo se geolocalizan los resultados sobre la avenida Don Bosco en Piura.

Figura 64

Detección de una falla de tipo piel de cocodrilo con Pavimenta2

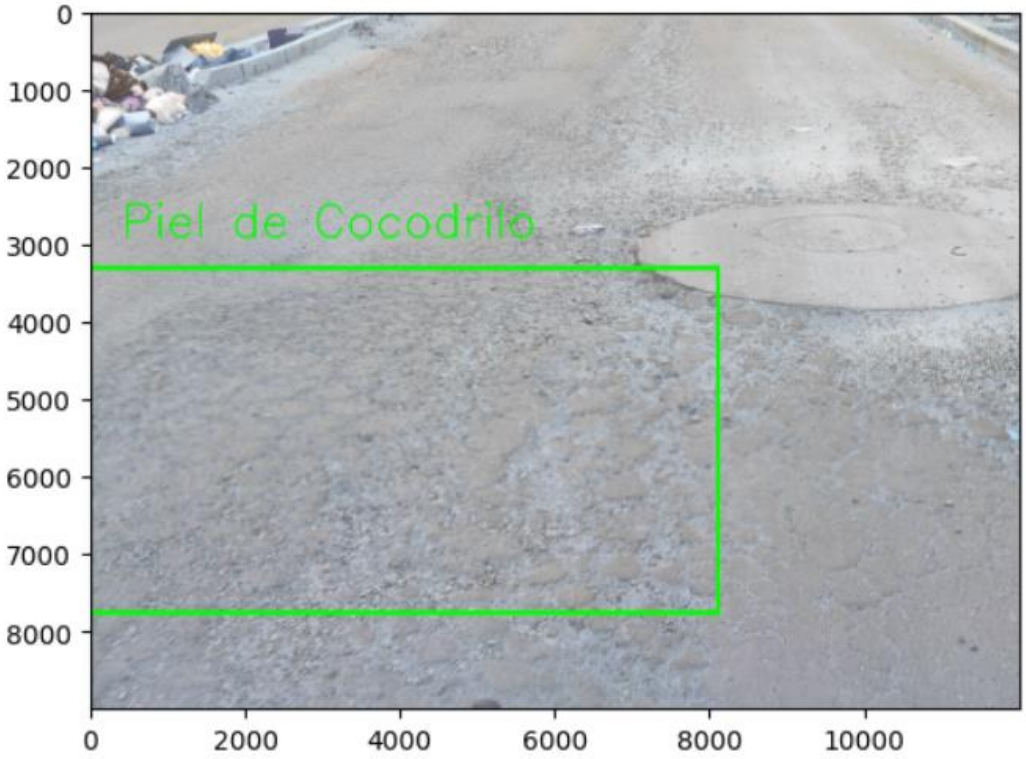
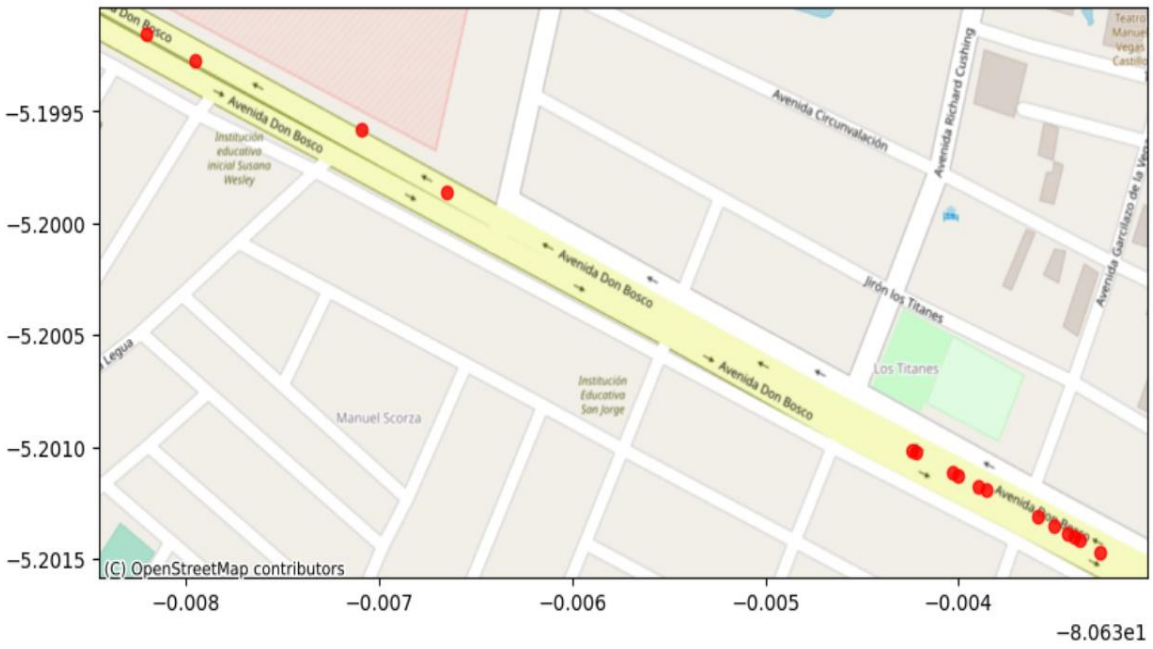


Figura 65

Detección de fallas usando Pavimenta2 en la Av. Don Bosco de Piura



Finalmente, en la Figura 66 se muestra una tabla donde se compara los hallazgos de nuestro software versus los encontrados por Pavimenta2, mostrando también el porcentaje de coincidencia de ambos; es decir, si nuestro software categoriza la falla de la misma manera que Pavimenta2, usando toda la base de datos mostrados en la Sección 3.1.1.1 tomados en la ciudad de Piura y que incluyen la importante avenida Don Bosco mostrada en la Figura 65.

Figura 66

Comparación entre software de la presente tesis versus Pavimenta2

	Tipo de fallas encontradas			
	Longitudinal		Piel de cocodrilo	
	Nmro de fallas	% de coincidencia	Nmro de fallas	% de coincidencia
Pavimenta2	23		18	
Software Tesis	25	92%	18	100%

Para reforzar los resultados de validación con el software Pavimenta2, se calculó el F-Score y se analizó el rendimiento de los modelos de detección de fallas en la vía pavimentada.

Dado que no se encontraron fallas de Fisuras Transversales en el tramo evaluado, el enfoque se centra únicamente en definir los términos para las clases de Fisuras Longitudinales y Piel de Cocodrilo en cada modelo:

Modelo de aprendizaje profundo:

- VP: Fisuras Longitudinales detectadas correctamente: 23
- FP: Fisuras Longitudinales detectadas incorrectamente: 0
- FN: Fisuras Longitudinales no detectadas: 2 (total reales 25 - VP 23)
- VN: Fisuras de Cocodrilo detectadas correctamente: 18

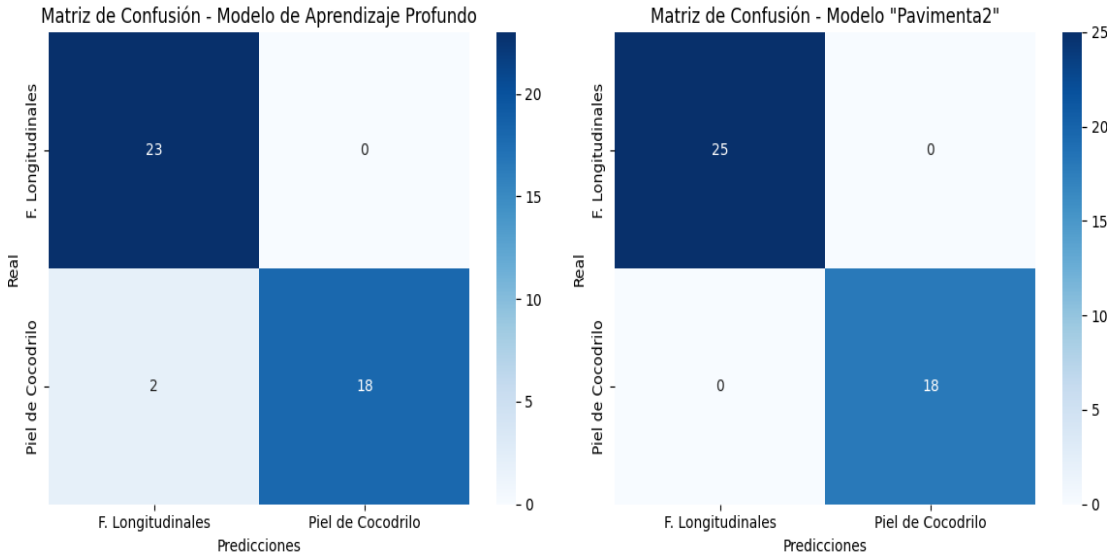
Modelo de "Pavimenta2":

- VP: Fisuras Longitudinales detectadas correctamente: 25
- FP: Fisuras Longitudinales detectadas incorrectamente: 0
- FN: Fisuras Longitudinales no detectadas: 0 (total reales 25 - VP 25)
- VN: Fisuras de Cocodrilo detectadas correctamente: 18

A partir de las definiciones, se graficó la matriz de confusión para ambos modelos en la Figura 67.

Figura 67

Matriz de confusión para el modelo de Aprendizaje Profundo y Pavimenta2



A partir de las fórmulas presentadas para las métricas de Precisión (Ecuación 16), Recall (Ecuación 17) y F1-Score (Ecuación 20), se reemplazaron los valores y se calcularon por clase cada una de las métricas de cada modelo.

Modelo de aprendizaje profundo:

- Fisuras Longitudinales

$$\text{Precisión}_{\textbf{Longitudinal}} = \frac{23}{23+0} = 1.0$$

$$\text{Recall}_{\textbf{Longitudinal}} = \frac{23}{23+2} = 0.92$$

$$\text{F1} - \text{Score}_{\textbf{Longitudinal}} = \frac{2 \times 1 \times 0.92}{1+0.92} = 0.96$$

- Piel de Cocodrilo

  En esta clase la precisión y recall también serán 1.0 y 1.0 respectivamente, así que el F1-Score en definitiva también será 1.0.

Modelo de "Pavimenta2":

- Fisuras Longitudinales y Piel de Cocodrilo

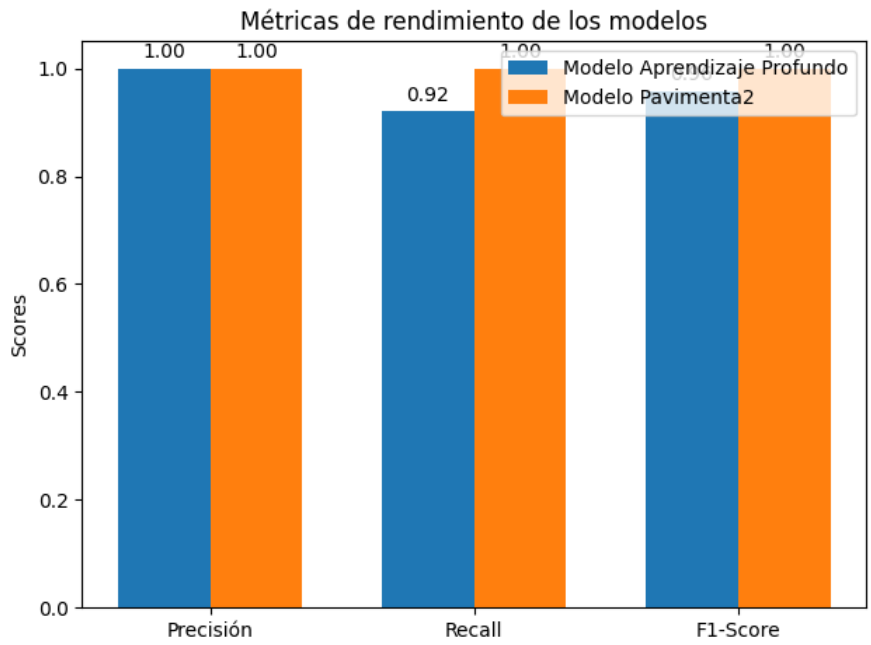
  Para ambas clases la precisión y recall serán de 1.0 y 1.0 respectivamente, así que el F1-Score en definitiva también será 1.0.

En resumen, en la Figura 68, se puede observar un gráfico del rendimiento de los modelos y se puede indicar lo siguiente:

- Precisión: ambos modelos tienen una precisión del 100% en la detección de Fisuras Longitudinales, lo que significa que, cuando un modelo predice que hay una fisura longitudinal, está correcto al 100%.
- Recall: el modelo de aprendizaje profundo tiene un recall de 92%, lo que significa que puede estar perdiendo algunas fisuras longitudinales (2 de 25), mientras que "Pavimenta2" tiene un recall perfecto, detectando todas las fisuras longitudinales en el tramo.
- F1-Score: el modelo de aprendizaje profundo tiene un F1-Score de aproximadamente 0.96, lo que indica un rendimiento muy alto, aunque no perfecto. El modelo "Pavimenta2" tiene un F1-Score perfecto de 1.0, lo que indica un rendimiento óptimo en esta tarea específica.

Figura 68

Métricas de rendimientos de los modelos



CAPÍTULO IV: CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

1. Se desarrolló un modelo de aprendizaje profundo entrenado con imágenes de la ciudad de Piura, el cual ha demostrado mejorar la precisión en el proceso de detección automática del estado de las superficies de pavimento flexible, alcanzando una precisión del 92.76% en la detección de fallas, como piel de cocodrilo, fisuras longitudinales y transversales.
2. La validación del modelo de aprendizaje profundo en comparación con el software Pavimenta2 confirma la métrica de evaluación F1-Score al 96 % para fisuras longitudinales y un 100% para piel de cocodrilo. Estos resultados respaldan la fiabilidad del modelo, ofreciendo seguridad a los usuarios.
3. El modelo de aprendizaje profundo alcanzó una alta precisión en solo 29 minutos y 16 segundos, superando las más de una hora requeridas por Pavimenta2. Estos resultados evidencian que el modelo no solo es preciso en la detección de fallas, sino que también ofrece una mayor eficiencia en tiempo.
4. Se desarrolló un modelo óptimo de detección automática, utilizando la red neuronal DenseNet como base y aplicando transferencia de aprendizaje. Lo que demuestra el gran potencial del aprendizaje profundo para desarrollar soluciones predictivas que automatizan la detección de fallas y reducen los tiempos de procesamiento.
5. El preprocesamiento de las imágenes recopiladas en campo, mediante la técnica de data augmentation, incrementó el número de imágenes necesarias para el entrenamiento del modelo. Esta técnica resultó fundamental para alcanzar los resultados del estudio, al aplicarse a los tres tipos de fallas definidos en la dimensión del conjunto de datos, lo que mejoró la precisión del modelo.
6. El uso de tarjetas gráficas de alto rendimiento, como la NVIDIA Tesla T4 y la AMD RTX 6900 XT, facilitó el procesamiento de una gran cantidad de fotografías y alcanzar modelos con alta precisión. Es relevante señalar que el desarrollo e implementación del modelo se llevó a cabo en un entorno de gabinete, optimizando los recursos.
7. El modelo computacional se desarrolló con el lenguaje de programación Python, lo que permitió automatizar el proceso de mapeo de las fallas detectadas en un mapa libre, como se realiza en los formatos GIS. Esta automatización optimiza la visualización de datos y facilita la toma de decisiones informadas para el mantenimiento y la intervención oportuna en la infraestructura vial.

4.2 RECOMENDACIONES

1. Integrar sistemas multisensoriales que combinen imágenes digitales, información de profundidad y datos de otras ciudades, para obtener una representación más completa del estado de las superficies del pavimento. Esto aumentará la precisión del modelo de aprendizaje profundo en la detección de fallas y fortalecerá la gestión de la infraestructura de pavimentos flexibles a nivel nacional.
2. Se propone realizar un estudio comparativo a gran escala sobre diferentes tipos de fallas de pavimento flexible y condiciones ambientales, utilizando modelos de aprendizaje profundo y otras herramientas de inteligencia artificial, como Pavimenta2. Este enfoque permitirá validar la consistencia de los resultados en diversos contextos, identificar áreas de mejora en el modelo y fortalecer la confianza de los usuarios en la herramienta, facilitando su adopción en la gestión de infraestructura vial.
3. Se recomienda implementar el modelo de aprendizaje profundo en el proceso de detección automática del estado de las superficies de pavimento flexible, ya que es preciso y eficiente. También se debe comparar continuamente con otros softwares de inteligencia artificial para optimizar la detección de fallas y elegir las mejores herramientas para la gestión de pavimentos.
4. Investigar y probar diferentes modelos de redes neuronales, incluyendo DenseNet, para mejorar la detección automática de fallas en pavimento flexible. Aplicar transferencia de aprendizaje para expandir el uso del aprendizaje profundo en la gestión de infraestructura vial en Piura y otras ciudades.
5. Se recomienda utilizar técnicas de aumento de datos en futuros estudios sobre preprocesamiento de imágenes para obtener conjuntos de datos más robustos. También se sugiere incluir grupos de control para fortalecer la confiabilidad de los resultados y permitir una evaluación más rigurosa del modelo.
6. Invertir en tarjetas gráficas avanzadas mejora el procesamiento de datos de aprendizaje profundo. Investigar métodos para procesamiento en tiempo real de sistemas de detección de grietas. Explorar algoritmos eficientes para dispositivos integrados, que permitan analizar rápidamente imágenes de la superficie del pavimento flexible capturadas por plataformas móviles, como drones o vehículos.
7. Conviene mejorar continuamente el modelo computacional, incorporando análisis y visualización avanzados, usando otros lenguajes y herramientas como Python. También se recomienda integrar el sistema con plataformas GIS comerciales para enriquecer la base de datos geoespacial y brindar más herramientas de gestión

REFERENCIAS BIBLIOGRÁFICAS

Amhaz, R., Chambon, S., Idier, J., & Baltazart, V. (2016). Automatic Crack Detection on Two-Dimensional Pavement Images: An Algorithm Based on Minimal Path Selection. *IEEE Transactions on Intelligent Transportation Systems, 17*(10), 2718-2729. https://doi.org/10.1109/TITS.2015.2477675

Angulo, A., Vega, J., Aguilar, L., Natraj, S., & Ochoa, G. (2019). Road Damage. *Advances in Soft Computing & Lecture Notes in Computer Science, Springer*, 3–14. https://doi.org/10.1007/978-3-030-33749-0_1

Arman, S., Hasan, M., Sadia, F., Khan Shakir, A., Sarker, K., & Anan, F. (2020). Detection and Classification of Road Damage Using R-CNN and Faster R-CNN: A Deep Learning Approach. *Cyber Security and Computer Science, 1*(325), 730–741. https://doi.org/10.1007/978-3-030-52856-0_58

Arya, D., Maeda, H., Kumar, G., Toshniwal, D., Mraz, A., Kashiyama, T., & Sekimoto, Y. (2021). Deep learning-based road damage detection and classification for multiple countries. *Automation in Construction, 132*(1), 1–19. https://doi.org/10.1016/j.autcon.2021.103935

avidi, B., Stephens, J., Kishk, S., Naughton, T., McDonald, J., & Isaac, A. (2003). Pilot For Automated Detection and Classification Of Road Surface Degradation Features. *Connecticut Department of Transportation*. https://api.semanticscholar.org/CorpusID:8521655

Ayenu-Prah, A., & Attoh-Okine, N. (2008). Evaluating Pavement Cracks with . *EURASIP*, 1–7. https://doi.org/10.1155/2008/861701

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*(12), 2481-2495. https://doi.org/10.1109/TPAMI.2016.2644615

Banco Interamericano de Desarrollo. (19 de 09 de 2022). *Pavimenta2*. https://blogs.iadb.org/transporte/es/pavimentados-acelerando-la-transformacion-digital-del-sector-transporte-en-america-latina-y-el-caribe/

Behera, A. (02 de 07 de 2021). *Convnets: What and how?* https://medium.com/analytics-vidhya/convnets-what-and-how-b3d9285edef4

Chambon, S., & Moliard, J. (2011). Automatic Road Pavement Assessment with Image Processing: Review and Comparison. *International Journal of Geophysics*(1), 1-20. https://doi.org/10.1155/2011/989354

Chatterjee, R. (2020). Fundamental concepts of artificial intelligence and its applications. *Journal of Mathematical Problems, Equations and Statistics, 1*(2), 13-24.

Chauhan, R., Ghanshala, K., & Joshi, R. (2018). Convolutional Neural Network (CNN) for Image Detection and Recognition. *First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, 278-282. https://doi.org/10.1109/ICSCCC.2018.8703316

Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*, 248-255. https://doi.org/10.1109/CVPR.2009.5206848

Dollár, P., & Zitnick, C. (2013). Structured Forests for Fast Edge Detection. *IEEE International Conference on Computer Vision*, 1841-1848. https://doi.org/10.1109/ICCV.2013.231

Dwivedi, P. (04 de 06 de 2019). *Understanding and coding a resnet in keras*. https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

Eisenbach, M., Stricker, R., Seichter, D., Amende, K., Debes, K., Sesselmann, M., Ebersbach, D., Stoeckert, U., & Gross, H. (2017). How to get pavement distress detection ready for deep learning? A systematic approach. *2017 International Joint*

Conference on Neural Networks (IJCNN), 2039-2047. https://doi.org/10.1109/IJCNN.2017.7966101

Esplana, K., & Pérez, C. (2021). *Influencia de la rigidez del pavimento en la condición superficial del pavi mento utilizando técnicas de inteligencia artificial en la vía nacional PE26B [Para optar el título profesional de Ingeniero Civil,Universidad Nacional de Huancavelica].* https://repositorio.unh.edu.pe/items/701b4a3c-04e4-4004-aa4e-24fb46b74e03

Everingham, M., Eslami, A., Van Gool, L., Williams, C., Winn, J., & Zisserman, A. (2015). The Pascal Visual Object Classes Challenge: A Retrospective. *Int J Comput Vis , 111*(1), 98–136. https://doi.org/10.1007/s11263-014-0733-5

Fan, H., Mei, X., Prokhorov, D., & Ling, H. (2018). Multi-level contextual rnns with attention model for scene labeling. *IEEE Transactions on Intelligent Transportation Systems, 19*(11), 3475-3485. https://doi.org/10.1109/TITS.2017.2775628

Fei, L. (s.f.). *Convolutional neural networks (cnns / convnets)*. https://cs231n.github.io/convolutional-networks/

Feng, C., Liu, M., Kao, C., & Lee, T. (2017). Deep Active Learning for Civil Infrastructure Defect Detection and Classification. *Computing in Civil Engineering* , 298-306. https://doi.org/10.1061/9780784480823.036

Fernandes, K., & Ciobanu, L. (2014). Pavement pathologies classification using graph-based features. *IEEE International Conference on Image Processing (ICIP)*, 793-797. https://doi.org/10.1109/ICIP.2014.7025159

Galdi, P., & Tagliaferri, R. (2018). Data Mining: Accuracy and Error Measures for Classification and Prediction. *Encyclopedia of Bioinformatics and Computational Biology*. https://doi.org/10.1016/B978-0-12-809633-8.20474-3

Goodfellow, I., Bengio, Y., & Courville, A. (2017). Deep learning: The MIT Press. *Genetic Programming and Evolvable Machines 2016, 19*. https://doi.org/10.1007/s10710-017-9314-z

Gui, R., Xu, X., Zhang, D., Lin, H., Pu, F., He, L., & Cao, M. (2018). A Component Decomposition Model for 3D Laser Scanning Pavement Data Based on High-Pass Filtering and Sparse Analysis. *Sensors, 18*(7), 2294. https://doi.org/10.3390/s18072294

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation.* Prentice Hall. https://dl.acm.org/doi/book/10.5555/541500

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Hernandez, R., Fernández, C., & Baptista, M. (2014). *Metodología de la Investigación.* Mc Graw Hill. https://doi.org/ISBN: 978-1-4562-2396-0

Hinton, G., Osindero, S., & Teh, Y. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation, 18*(7), 1527-1554. https://doi.org/10.1162/neco.2006.18.7.1527

Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *Computer Vision and Pattern Recognition*. https://doi.org/10.48550/arXiv.1704.04861

Hu, Y., & Zhao, C. (2009). A novel LBP based methods for pavement crack detection. *Image Processing and Communications, 14*, 5-12. https://api.semanticscholar.org/CorpusID:61244503

Huang, J., Liu, W., & Sun, X. (2014). A Pavement Crack Detection Method Combining 2D with 3D Information Based on Dempster-Shafer Theory. *Computer-Aided Civil and Infrastructure Engineering, 29*(4), 299-313. https://doi.org/10.1111/mice.12041

Javidi, B., Stephens, J., Kishk, S., Naughton, T., McDonald, J., & Isaac, A. (2003). Pilot for automated detection and classification of road surface degradation features.

Connecticut Department of Transportation. https://api.semanticscholar.org/CorpusID:8521655

Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision, 1*(4), 321–331. https://doi.org/10.1007/BF00133570

Kaul, V., Yezzi, A., & Tsai, Y. (2012). Detecting Curves with Unknown Endpoints and Arbitrary Topology Using Minimal Paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 34*(10), 1952-1965. https://doi.org/10.1109/TPAMI.2011.267

Krizhevsky, A., Sutskever, I., & Hinton, G. (2017). ImageNet classification with deep convolutional neural networks. *Association for Computing Machinery, 60*(6), 0001-0782. https://doi.org/10.1145/3065386

Lao, Y., Rivas, A., Pérez, M., & Marrero, F. (2017). Procedimiento para el pronóstico de la demanda mediante redes neuronales artificiales / Procedure for forecasting demand by using artificial neural networks. *Ciencias Holguin, 23*, 43-59. https://www.researchgate.net/publication/313238723

Lecun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation, 1*(4), 541-551. https://doi.org/10.1162/neco.1989.1.4.541

Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. (2014). Microsoft COCO: Common Objects in Context. *Lecture Notes in Computer Science, 8693*, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

Long, J. S. (2017). Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*, 640-651. https://doi.org/10.1109/TPAMI.2016.2572683

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., & Omata, H. (2018). Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images. *Computer-Aided Civil and Infrastructure Engineering, 33*(12), 1127–1141. https://doi.org/10.1111/mice.12387

Majidifard, H., Jin, P., Adu-Gyamfi, Y., & Buttlar, W. (2020). Pavement Image Datasets: A New Benchmark Dataset to Classify and Densify Pavement Distresses. *Transportation Research Record, 2674*(2), 328-339. https://doi.org/10.1177/0361198120907283

Maniat, M., Camp, C., & Kashani, A. (2021). Deep learning-based visual crack detection using Google Street View images. *Neural Computing and Applications, 33*(1), 14565-14582. https://doi.org/10.1007/s00521-021-06098-0

Manli, S., Zhanjie, S., Xiaoheng, J., Jing, P., & Yanwei, P. (2017). Learning Pooling for Convolutional Neural Network. *Neurocomputing, 224*(1), 96-104. https://doi.org/10.1016/j.neucom.2016.10.049

Manzanares, A. (2019). *Detector de Baches con Deep Learning [Tesis de Maestría, Escuela Superior Politécnica].* Repositorio Institucional. http://hdl.handle.net/10230/42402

Maode, Y., Shao-bo, B., Kun, X., & Yuyao, H. (2007). Pavement Crack Detection and Analysis for High-grade Highway. *8th International Conference on Electronic Measurement and Instruments, 4*, 548-552. https://doi.org/10.1109/ICEMI.2007.4351202

Ministerio de Transportes y Comunicaciones. (2014). *Manual de carreteras Suelos,Geología, Geotecnia y Pavimentos.* Dirección General de Caminos y Ferrocarriles.

Ministerio de Transportes y Comunicaciones. (2018). *Manual de Carreteras - Mantenimiento o Conservación Vial.* Dirección General de Caminos y Ferrocarriles.

Montalvo, A. (2021). *Sistema Electrónico de Detección de Baches y Topes para Automóviles [Tesis de Maestría, Instituto Tecnológico de Celaya, México].* https://doi.org/10.13140/RG.2.2.34688.15362

Moolchandani, D., Kumar, A., & Sarangi, S. (2021). Accelerating CNN Inference on ASICs: A Survey. *Journal of Systems Architecture, 113*(1), 1383-7621. https://doi.org/10.1016/j.sysarc.2020.101887

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective.* The MIT.

Nguyen, H., Nguyen, L., & Sidorov, D. (2016). A Robust Approach for Road Pavement Defects Detection and Classification. *Journal of Computational and Engineering Mathematics, 3*(3), 40–52. https://doi.org/10.14529/jcem160305

Ortega, J. (2021). *Aprendizaje profundo para la detección automática de fisuras de hormigón usando redes neuronales convolucionales [Tesis de Maestría, Universitat Politècnica de València - España].* Repositorio Institucional. https://riunet.upv.es/handle/10251/174954

Ouyang, W., & Xu, B. (2013). Pavement cracking measurements using 3D laser-scan images. *Measurement Science and Technology, 24*(10), 1–9. https://doi.org/10.1088/0957-0233/24/10/105204

Pauly, L., Peel, H., Luo, S., Hogg, D., & Fuentes, R. (2017). Deeper Networks for Pavement Crack Detection. *International Symposium on Automation and Robotics in Construction*. https://doi.org/10.22260/ISARC2017/0066

Provias Nacional. (2020). *Memoria Anual 2019.* Ministerio de Transportes y Comunicaciones (MTC).

Quintana, M., Torres, J., & Menéndez, J. (2016). A Simplified Computer Vision Sys- tem for Road Surface Inspection and Maintenance. *IEEE Transactions on Intelligent Transportation Systems, 17*(3), 608-619. https://doi.org/10.1109/TITS.2015.2482222

Reidl-Martínez, L. (2012). Marco conceptual en el proceso de investigación. *Inves-, 1*(3), 146–151. http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2007-50572012000300007&lng=es&nrm=iso

Ríos, N., Bacca, B., Caicedo, E., & Orobio, A. (2020). Revisión de métodos para la clasificación de fallas superficiales en pavimentos flexibles. *Ciencia e Ingeniería Neogranadina, 30*(2), 109–127. https://doi.org/10.18359/rcin.4385

Rodríguez, F. (2020). *Sistema automático para la detección y clasificación de grietas en pavimentos [Tesis de Doctorado, Universidad de Cordova].* Repositorio Institucional. http://hdl.handle.net/10396/20499

Ros, A. (2019). *Sistema de percepción de elementos viarios usando técnicas de visión por computador para aplicación en conducción autónoma [Tesis de Maestría, Universidad Politécnica de Cartagena-Cordova].* Respositorio Institucional. http://hdl.handle.net/10317/8286

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach.* Pearson.

Saleh, Z. (2019). Artificial Intelligence Definition, Ethics and Standards. *Electronics and Communications: Law, Standards and Practice*. https://www.researchgate.net/publication/332548325_Artificial_Intelligence_Definition_Ethics_and_Standards

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2019). Recent Advances in Recurrent Neural Networks. *Neural and Evolutionary Computing, 3*(1), 1–21. https://www.researchgate.net/publication/322243714

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *IEEE Conference on Computer Vision and Pattern Recognition, 4*(1), 4510-4520. https://doi.org/arXiv:1801.04381v4

Schmugge, S., Rice, L., Lindberg, J., Grizzi, R., Joffe, C., & Shin, M. (2017). Crack Segmentation by Leveraging Multiple Frames of Varying Illumination. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1045-1053. https://doi.org/10.1109/WACV.2017.121

Sensio, J. (12 de 09 de 2020). *Transfer Learning en Redes Convolucionales*. https://juansensio.com/blog/044_cnn_transfer_learning

Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z. (2016). Automatic Road Crack Detection Using Random Structured Forests. *IEEE Transactions on Intelligent Transportation Systems, 17*(12), 3434-3445. https://doi.org/10.1109/TITS.2016.2552248

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818-2826. https://doi.org/10.1109/CVPR.2016.308

Tsang, S. (2018). *Review: Densenet — dense convolutional network (Image Classification)*. https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

Tsuchiya, H., Fukui, S., Iwahori, Y., Yoshitsugu, H., Achariyaviriya, H., & Kijsirikul, B. (2019). A Method of Data Augmentation for Classifying Road Damage Considering Influence on Classification Accuracy. *Procedia Computer Science, 159*(1), 1449-1458. https://doi.org/10.1016/j.procs.2019.09.315

Universidad Politécnica De Madrid. (2022). *Introducción al Aprendizaje Automático*. https://dcain.etsin.upm.es/~carlos/bookAA/introAA.html

Yan, W., & Yuan, X. (2018). A low-cost video-based pavement distress screening system for low-volume roads. *Journal of Intelligent Transportation Systems Technology Planning and Operations, 22*(1), 376 - 389. https://doi.org/10.1080/15472450.2017.1366320

Yang, F., Zhang, L., Yu, S., Prokhorov, D., Mei, X., & Ling, H. (2019). Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection. *IEEE Transactions on Intelligent Transportation Systems*, 1-11. https://doi.org/10.1109/TITS.2019.2910595

Yang, N., Li, R., & Ma., Y. y. (2022). An Efficient Method for Detecting Asphalt Pavement Cracks and Sealed Cracks Based on a Deep Data-Driven Model. *Applied Sciences, 12*(19), 1–23. https://doi.org/10.3390/app121910089

Zhang, L., Yang, F., Zhang, Y., & Zhu, Y. (2016). Road crack detection using deep convolutional neural network. *IEEE International Conference on Image Processing (ICIP)*, 3708-3712. https://doi.org/10.1109/ICIP.2016.7533052

Zhou, J., Huang, P., & Chiang, F.-P. (2005). Wavelet-Based Pavement Distress Classification. *Transportation Research Record, 1940*(1), 89–98. https://doi.org/10.3141/1940-11

Zúñiga, Y. (s.f.). *Deep learning para la detección de fallas en pavimentos de una zona del distrito de Villa María del Triunfo 2022[Para optar el título profesional de Ingeniero de Sistemas, Universidad César Vallejo-Lima].* Respositorio Institucional. https://hdl.handle.net/20.500.12692/93124

ANEXOS

Anexo 1: Matriz de Consistencia

Título: DETECCIÓN AUTOMÁTICA DEL ESTADO DE SUPERFICIES DE PAVIMENTO FLEXIBLE EN LA CIUDAD DE PIURA UTILIZANDO APRENDIZAJE PROFUNDO.					
Problema de Estudio	**Problemas**	**Objetivos**	**Hipótesis**	**Variables y dimensiones**	**Metodología**
¿Cómo mejorar el proceso de detección automática del estado de las superficies de pavimento flexible en la ciudad de Piura de forma que sea eficiente en precisión?	**Problema general** ¿Cómo mejorar la precisión en el proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura mediante el uso de aprendizaje profundo?	**Objetivo general** Determinar un modelo de Aprendizaje Profundo que permita mejorar la precisión en el proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura.	**Hipótesis general** El uso del modelo de Aprendizaje Profundo mejora significativamente la precisión en el proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura, permitiendo detectar automáticamente los diferentes tipos de fallas con una precisión del 92.76 %.	**Variable independiente** Modelo de Aprendizaje Profundo **DIMENSIONES:** - Conjunto de datos (imágenes) de acuerdo con el tipo de grietas en el pavimento flexible a evaluar (Tipo de falla según normativa nacional: Fallas del tipo fisura longitudinal, fisura transversal y piel de cocodrilo). - Algoritmos de aprendizaje profundo de detección (redes preentrenadas): DenseNet-BC, Inception versión 3, ResNet-50 y Mobilenet versión 2.	**Tipo de investigación:** por enfoque, es cuantitativa porque busca analizar la relación entre las variables de manera objetiva y numérica. **Nivel de investigación:** la investigación alcanza un nivel aplicado, ya que se enfoca en la aplicación práctica del aprendizaje profundo. **Métodos:** según el método de investigación, el tipo de inferencia es hipotético-deductivo. **Diseño:** tiene como base las características de un diseño experimental. Sin embargo, en un marco de control por parte de la investigación, se clasifica como preexperimental, al no contar con un grupo de control, lo que limita la validez interna.
	Problemas específicos • **Problema específico 1.** ¿Cómo desarrollar un modelo de Aprendizaje Profundo que mejore la precisión del proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura?	**Objetivos específicos** • **Objetivo específico 1.** Entrenar un modelo de Aprendizaje Profundo para mejorar la precisión del proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura.	**Hipótesis especificas** • **Hipótesis Específica 1.** El modelo de Aprendizaje Profundo mejora el proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura al detectar las fallas con una precisión igual al 92.76 %.	**Variable dependiente** Precisión en el proceso de detección automática del estado de superficies de pavimento flexible**.** **DIMENSIONES:** - Detección y clasificación automática del tipo de falla. (Porcentaje de acierto de patrones o características al ejecutar el modelo con la métrica de evaluación:	**Población y muestra**: La población de esta investigación serían todas las imágenes de las vías de pavimento flexible de la ciudad de Piura, extraídas de los videos grabados en las diferentes avenidas. La muestra final consta de 3312 imágenes, distribuidas de la siguiente manera: - Fisura longitudinal: 1104 imágenes. - Fisura transversal: 1104 imágenes. - Piel de cocodrilo: 1104 imágenes.

	• **Problema específico 2.** ¿Cómo validar la aplicación del modelo de Aprendizaje Profundo que es utilizado en la mejora de la precisión del proceso de detección automática de estados en superficies de pavimento flexible de la ciudad de Piura?	• **Objetivo específico 2.** Validar la aplicación del modelo de Aprendizaje Profundo para mejorar la precisión del proceso de detección automática de fallas en superficies de pavimento flexible en la ciudad de Piura en comparación con Pavimenta2. .	• **Hipótesis Específica 2.** El modelo de aprendizaje profundo tendrá una métrica de evaluación F1-Score igual o superior al 96% en comparación con el software Pavimenta2, lo que valida la mejora de precisión del proceso de detección automática del estado de superficies de pavimento flexible en la ciudad de Piura.	accuracy, precisión por clase, recall y F1-Score.) $Precisión\ (accuracy) = \frac{número\ de\ predicciones\ correctas}{número\ total\ de\ predicciones}$ $Precisión_{clase} = \frac{VP}{VP + FP}$ $Recall = \frac{VP}{VP + FN}$ $F_1 = 2 \times \frac{Precisión_{clase} \times Recall}{(Precisión_{clase}) + Recall}$	**Técnicas e instrumentos de recolección de datos:** Se recolectó la data mediante la técnica de observación se usó sistemas de escaneo por área (fotografías) y video, capturados con un smartphone de alta gama (Xiaomi 11T Pro) desde un vehículo modelo CHERY Q 22. Para registrar la cantidad de imágenes seleccionadas, se utilizó un formato específico que se muestra en el anexo 4. **Técnicas e instrumentos de análisis y procesamiento de datos:** Preprocesamiento de imágenes para ajustar, mejorar la calidad y aumentar el conjunto de datos. Se automatizo con script ejecutados en Python y se usó el software de edición de imágenes Adobe Photoshop. Para realizar tareas específicas de procesamiento de imágenes y videos, como la detección y clasificación de objetos, se empleó ImageAI, una biblioteca que complementa a TensorFlow para el modelado en aprendizaje profundo. La métrica de precisión de prueba (test accuracy) se presenta mediante gráficos lineales. Se valida el modelo de aprendizaje profundo al comparar las métricas de evaluación Precisión por clase, Recall y F1-Score de la detección automática de fallas con el sotware Pavimenta2.

Fuente: Elaboración propia.

Anexo 2: Matriz de Operacionalización de Variables.

<table>
<tr><th rowspan="2">VARIABLE</th><th rowspan="2">DEFINICIÓN CONCEPTUAL</th><th rowspan="2">DEFINICIÓN OPERACIONAL</th><th colspan="3">OPERACIONALIZACIÓN</th><th rowspan="2">ESCALA DE MEDICIÓN</th></tr>
<tr><th>DIMENSIONES</th><th>INDICADORES</th><th>INDICES Y/O ITEMS</th></tr>
<tr><td rowspan="2"><u>Variable independiente:</u>
Modelo de Aprendizaje Profundo</td><td rowspan="2">El modelo de aprendizaje profundo es un algoritmo de aprendizaje automático que utiliza redes neuronales artificiales con múltiples capas para aprender automáticamente patrones a partir de datos que le permiten realizar predicciones o clasificaciones precisas en varios dominios, como el reconocimiento de imágenes, lo que propicia la detección automática de objetos, con un alto grado de precisión.</td><td rowspan="2">En la práctica, para desarrollar un modelo de aprendizaje profundo se requiere de redes neuronales artificiales con múltiples capas y la selección de un conjunto adecuado de datos de entrenamiento, en este caso de imágenes. La medición se realizará a base de la métrica de precisión del nuevo algoritmo de aprendizaje automático obtenido.</td><td>Red neuronal pre-entrenada (Algoritmo de aprendizaje profundo de detección automática)</td><td>Múltiples capas que tiene la red neuronal seleccionada DenseNet</td><td></td><td rowspan="2">- Cuantitativos: De la estadística computacional se obtiene la precisión del nuevo modelo (porcentaje de aciertos de las predicciones que vaya a realizar el algoritmo).</td></tr>
<tr><td>Data de imágenes de acuerdo a su tipo de falla categorizada</td><td>Cantidad de imágenes por su tipo de falla categorizada.</td><td></td></tr>
<tr><td><u>Variable dependiente</u>
La Precisión en el proceso de Detección automática de estados en superficies de pavimento flexible</td><td>Capacidad de los algoritmos de aprendizaje automático para identificar correctamente los diferentes tipos de fallas en la superficie del pavimento. Una alta precisión de ello permite una evaluación más fiable y precisa en el proceso detección automática del estado de superficies de pavimentos y la planificación de los trabajos de mantenimiento y reparación necesarios, lo que contribuye a mejorar la seguridad vial y reducir los costos a largo plazo.</td><td>En la práctica, para detectar automáticamente los estados en pavimentos flexibles, es común evaluar la precisión de los modelos de aprendizaje profundo. Esto se logra mediante la utilización de conjuntos de datos de prueba para determinar su capacidad para detectar y clasificar de manera adecuada diferentes tipos de fallas del pavimento. Por tanto, la medición se realizará al comparar la precisión de diferentes modelos, con el fin de validar el modelo propuesto para su uso en aplicaciones reales de pavimentación y que permita realizar una planificación eficiente en las operaciones de mantenimiento y reparación.</td><td>Tipos de fallas detectadas y clasificadas automáticamente en la superficie del pavimento flexible.</td><td>Cantidad por el tipo de fallas detectadas y clasificadas automáticamente de la superficie del pavimento flexible.</td><td></td><td>Cuantitativos: De la estadística descriptiva se obtiene el porcentaje de coincidencia al detectar y clasificar el tipo de falla.</td></tr>
</table>

Fuente: Elaboración propia.

Anexo 3: Resultados de las 100 Épocas de Entrenamiento.

En este anexo se detalla como prueba del proceso de iteraciones de épocas durante el entrenamiento de uno de los modelos de aprendizaje profundo, el cual logra una precisión final del 91.40%. Esta instancia representa un avance en el proceso de entrenamiento y la mejora progresiva del modelo hacia su versión final óptima.

```
Training with GPU
==================================================
Epoch 1/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.16it/s]
train Loss: 6.3633 Accuracy: 0.4926
100%|██████████| 14/14 [00:03<00:00,  3.52it/s]
test Loss: 1.9579 Accuracy: 0.5633
Epoch 2/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 1.5207 Accuracy: 0.4949
100%|██████████| 14/14 [00:03<00:00,  3.85it/s]
test Loss: 2.8809 Accuracy: 0.4751
Epoch 3/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.9098 Accuracy: 0.5023
100%|██████████| 14/14 [00:03<00:00,  3.81it/s]
test Loss: 0.9585 Accuracy: 0.4683
Epoch 4/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.8326 Accuracy: 0.5730
100%|██████████| 14/14 [00:03<00:00,  3.87it/s]
test Loss: 0.9642 Accuracy: 0.5633
Epoch 5/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.8486 Accuracy: 0.6008
100%|██████████| 14/14 [00:03<00:00,  3.81it/s]
test Loss: 4.7811 Accuracy: 0.4932
Epoch 6/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.6782 Accuracy: 0.6546
100%|██████████| 14/14 [00:03<00:00,  3.80it/s]
test Loss: 3.1220 Accuracy: 0.5882
Epoch 7/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.6149 Accuracy: 0.7242
100%|██████████| 14/14 [00:03<00:00,  3.76it/s]
test Loss: 6.6994 Accuracy: 0.6403
Epoch 8/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.6156 Accuracy: 0.6818
100%|██████████| 14/14 [00:04<00:00,  3.43it/s]
test Loss: 2.0826 Accuracy: 0.7398
```

```
Epoch 9/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.26it/s]
train Loss: 0.5505 Accuracy: 0.7469
100%|██████████| 14/14 [00:04<00:00,  3.30it/s]
test Loss: 1.3169 Accuracy: 0.5950
Epoch 10/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.26it/s]
train Loss: 0.5723 Accuracy: 0.7469
100%|██████████| 14/14 [00:03<00:00,  3.53it/s]
test Loss: 2.7108 Accuracy: 0.5882
Epoch 11/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.5806 Accuracy: 0.7571
100%|██████████| 14/14 [00:03<00:00,  3.85it/s]
test Loss: 1.4011 Accuracy: 0.6176
Epoch 12/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.5356 Accuracy: 0.7690
100%|██████████| 14/14 [00:03<00:00,  3.82it/s]
test Loss: 5.7192 Accuracy: 0.5520
Epoch 13/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.5164 Accuracy: 0.7724
100%|██████████| 14/14 [00:03<00:00,  3.85it/s]
test Loss: 1.5583 Accuracy: 0.5950
Epoch 14/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.5262 Accuracy: 0.7724
100%|██████████| 14/14 [00:03<00:00,  3.86it/s]
test Loss: 1.0588 Accuracy: 0.6335
Epoch 15/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.4971 Accuracy: 0.7865
100%|██████████| 14/14 [00:03<00:00,  3.84it/s]
test Loss: 1.4359 Accuracy: 0.5882
Epoch 16/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.4689 Accuracy: 0.8103
100%|██████████| 14/14 [00:03<00:00,  3.85it/s]
test Loss: 1.0837 Accuracy: 0.6833
Epoch 17/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.4775 Accuracy: 0.7984
100%|██████████| 14/14 [00:03<00:00,  3.59it/s]
test Loss: 1.1848 Accuracy: 0.6629
Epoch 18/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.5237 Accuracy: 0.7984
100%|██████████| 14/14 [00:04<00:00,  3.32it/s]
test Loss: 1.2480 Accuracy: 0.7059
Epoch 19/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.4548 Accuracy: 0.8126
100%|██████████| 14/14 [00:04<00:00,  3.32it/s]
test Loss: 1.5635 Accuracy: 0.6516
Epoch 20/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.4538 Accuracy: 0.8109
100%|██████████| 14/14 [00:03<00:00,  3.58it/s]
test Loss: 1.6898 Accuracy: 0.6629
Epoch 21/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.4717 Accuracy: 0.8126
100%|██████████| 14/14 [00:03<00:00,  3.86it/s]
test Loss: 0.9314 Accuracy: 0.7285
Epoch 22/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.4647 Accuracy: 0.8148
100%|██████████| 14/14 [00:03<00:00,  3.79it/s]
test Loss: 1.1464 Accuracy: 0.6448
Epoch 23/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.4430 Accuracy: 0.8222
100%|██████████| 14/14 [00:03<00:00,  3.81it/s]
test Loss: 0.9778 Accuracy: 0.6719
Epoch 24/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.4155 Accuracy: 0.8398
100%|██████████| 14/14 [00:03<00:00,  3.83it/s]
test Loss: 1.1300 Accuracy: 0.6719
```

```
Epoch 25/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.3902 Accuracy: 0.8465
100%|██████████| 14/14 [00:03<00:00,  3.85it/s]
test Loss: 0.9463 Accuracy: 0.6810
Epoch 26/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.4449 Accuracy: 0.8143
100%|██████████| 14/14 [00:03<00:00,  3.84it/s]
test Loss: 0.8128 Accuracy: 0.6448
Epoch 27/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.4111 Accuracy: 0.8301
100%|██████████| 14/14 [00:03<00:00,  3.62it/s]
test Loss: 0.8294 Accuracy: 0.6290
Epoch 28/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.4033 Accuracy: 0.8284
100%|██████████| 14/14 [00:04<00:00,  3.34it/s]
test Loss: 0.7121 Accuracy: 0.6584
Epoch 29/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.26it/s]
train Loss: 0.4205 Accuracy: 0.8301
100%|██████████| 14/14 [00:04<00:00,  3.28it/s]
test Loss: 1.0635 Accuracy: 0.6493
Epoch 30/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.3817 Accuracy: 0.8409
100%|██████████| 14/14 [00:03<00:00,  3.57it/s]
test Loss: 0.9384 Accuracy: 0.6742
Epoch 31/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.3435 Accuracy: 0.8675
100%|██████████| 14/14 [00:03<00:00,  3.81it/s]
test Loss: 0.6034 Accuracy: 0.7443
Epoch 32/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.3737 Accuracy: 0.8590
100%|██████████| 14/14 [00:03<00:00,  3.83it/s]
test Loss: 0.6478 Accuracy: 0.7353
```

```
Epoch 33/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.3448 Accuracy: 0.8607
100%|██████████| 14/14 [00:03<00:00,  3.79it/s]
test Loss: 1.6448 Accuracy: 0.7308
Epoch 34/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.3478 Accuracy: 0.8652
100%|██████████| 14/14 [00:03<00:00,  3.88it/s]
test Loss: 0.7605 Accuracy: 0.7489
Epoch 35/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.3695 Accuracy: 0.8460
100%|██████████| 14/14 [00:03<00:00,  3.86it/s]
test Loss: 1.2569 Accuracy: 0.6833
Epoch 36/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.3093 Accuracy: 0.8828
100%|██████████| 14/14 [00:03<00:00,  3.77it/s]
test Loss: 0.9786 Accuracy: 0.7081
Epoch 37/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.3195 Accuracy: 0.8766
100%|██████████| 14/14 [00:04<00:00,  3.47it/s]
test Loss: 0.8189 Accuracy: 0.7353
Epoch 38/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.3314 Accuracy: 0.8590
100%|██████████| 14/14 [00:04<00:00,  3.26it/s]
test Loss: 0.5242 Accuracy: 0.7489
Epoch 39/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.26it/s]
train Loss: 0.2913 Accuracy: 0.8884
100%|██████████| 14/14 [00:04<00:00,  3.46it/s]
test Loss: 0.6246 Accuracy: 0.7579
Epoch 40/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.3083 Accuracy: 0.8800
100%|██████████| 14/14 [00:03<00:00,  3.76it/s]
test Loss: 0.6716 Accuracy: 0.7896
```

```
Epoch 41/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.2959 Accuracy: 0.8817
100%|██████████| 14/14 [00:03<00:00,  3.87it/s]
test Loss: 0.6663 Accuracy: 0.7036
Epoch 42/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.2776 Accuracy: 0.8901
100%|██████████| 14/14 [00:03<00:00,  3.85it/s]
test Loss: 0.4444 Accuracy: 0.7647
Epoch 43/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.2816 Accuracy: 0.8901
100%|██████████| 14/14 [00:03<00:00,  3.80it/s]
test Loss: 0.4604 Accuracy: 0.8077
Epoch 44/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.3155 Accuracy: 0.8669
100%|██████████| 14/14 [00:03<00:00,  3.87it/s]
test Loss: 0.4691 Accuracy: 0.7557
Epoch 45/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.2853 Accuracy: 0.8766
100%|██████████| 14/14 [00:03<00:00,  3.82it/s]
test Loss: 0.6982 Accuracy: 0.7353
Epoch 46/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.2596 Accuracy: 0.9015
100%|██████████| 14/14 [00:03<00:00,  3.56it/s]
test Loss: 0.6281 Accuracy: 0.8484
Epoch 47/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.2569 Accuracy: 0.8975
100%|██████████| 14/14 [00:04<00:00,  3.22it/s]
test Loss: 0.5595 Accuracy: 0.7624
Epoch 48/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.2423 Accuracy: 0.9020
100%|██████████| 14/14 [00:04<00:00,  3.47it/s]
test Loss: 0.3746 Accuracy: 0.8235
```

```
Epoch 49/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.2842 Accuracy: 0.8901
100%|██████████| 14/14 [00:03<00:00,  3.86it/s]
test Loss: 0.6910 Accuracy: 0.6991
Epoch 50/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.2768 Accuracy: 0.8822
100%|██████████| 14/14 [00:03<00:00,  3.84it/s]
test Loss: 0.5284 Accuracy: 0.7986
Epoch 51/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.2511 Accuracy: 0.8992
100%|██████████| 14/14 [00:03<00:00,  3.84it/s]
test Loss: 0.6174 Accuracy: 0.7670
Epoch 52/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.2532 Accuracy: 0.8924
100%|██████████| 14/14 [00:03<00:00,  3.85it/s]
test Loss: 0.4512 Accuracy: 0.7964
Epoch 53/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.2491 Accuracy: 0.9009
100%|██████████| 14/14 [00:03<00:00,  3.79it/s]
test Loss: 0.8221 Accuracy: 0.6629
Epoch 54/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.2846 Accuracy: 0.8845
100%|██████████| 14/14 [00:03<00:00,  3.83it/s]
test Loss: 0.4613 Accuracy: 0.8009
Epoch 55/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.2329 Accuracy: 0.9026
100%|██████████| 14/14 [00:03<00:00,  3.57it/s]
test Loss: 0.3514 Accuracy: 0.8100
Epoch 56/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.2312 Accuracy: 0.9071
100%|██████████| 14/14 [00:04<00:00,  3.35it/s]
test Loss: 0.4807 Accuracy: 0.8145
```

```
Epoch 57/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.26it/s]
train Loss: 0.2386 Accuracy: 0.9049
100%|██████████| 14/14 [00:04<00:00,  3.34it/s]
test Loss: 1.0924 Accuracy: 0.7240
Epoch 58/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.2439 Accuracy: 0.8969
100%|██████████| 14/14 [00:03<00:00,  3.63it/s]
test Loss: 0.4282 Accuracy: 0.7896
Epoch 59/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.2589 Accuracy: 0.8924
100%|██████████| 14/14 [00:03<00:00,  3.78it/s]
test Loss: 0.6526 Accuracy: 0.7127
Epoch 60/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.3383 Accuracy: 0.8601
100%|██████████| 14/14 [00:03<00:00,  3.85it/s]
test Loss: 0.5950 Accuracy: 0.7262
Epoch 61/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.2567 Accuracy: 0.9015
100%|██████████| 14/14 [00:03<00:00,  3.86it/s]
test Loss: 0.3962 Accuracy: 0.8145
Epoch 62/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.2330 Accuracy: 0.9054
100%|██████████| 14/14 [00:03<00:00,  3.83it/s]
test Loss: 0.3546 Accuracy: 0.8824
Epoch 63/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.2130 Accuracy: 0.9173
100%|██████████| 14/14 [00:03<00:00,  3.86it/s]
test Loss: 0.3520 Accuracy: 0.8597
Epoch 64/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.2131 Accuracy: 0.9162
100%|██████████| 14/14 [00:03<00:00,  3.69it/s]
test Loss: 0.2876 Accuracy: 0.8710
```

```
Epoch 65/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.2063 Accuracy: 0.9145
100%|██████████| 14/14 [00:04<00:00,  3.44it/s]
test Loss: 0.9720 Accuracy: 0.7557
Epoch 66/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.26it/s]
train Loss: 0.2152 Accuracy: 0.9083
100%|██████████| 14/14 [00:04<00:00,  3.23it/s]
test Loss: 0.4382 Accuracy: 0.8009
Epoch 67/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.1973 Accuracy: 0.9230
100%|██████████| 14/14 [00:03<00:00,  3.54it/s]
test Loss: 0.4158 Accuracy: 0.8213
Epoch 68/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.2200 Accuracy: 0.9122
100%|██████████| 14/14 [00:03<00:00,  3.78it/s]
test Loss: 0.4414 Accuracy: 0.7828
Epoch 69/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.2272 Accuracy: 0.9224
100%|██████████| 14/14 [00:03<00:00,  3.89it/s]
test Loss: 0.4124 Accuracy: 0.8054
Epoch 70/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.2227 Accuracy: 0.9105
100%|██████████| 14/14 [00:03<00:00,  3.83it/s]
test Loss: 0.5238 Accuracy: 0.7738
Epoch 71/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.2218 Accuracy: 0.9088
100%|██████████| 14/14 [00:03<00:00,  3.87it/s]
test Loss: 0.9659 Accuracy: 0.7240
Epoch 72/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.2088 Accuracy: 0.9202
100%|██████████| 14/14 [00:03<00:00,  3.87it/s]
test Loss: 0.3535 Accuracy: 0.8190
```

```
Epoch 73/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.2136 Accuracy: 0.9134
100%|██████████| 14/14 [00:03<00:00,  3.84it/s]
test Loss: 0.2896 Accuracy: 0.8620
Epoch 74/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.2091 Accuracy: 0.9179
100%|██████████| 14/14 [00:03<00:00,  3.65it/s]
test Loss: 0.3664 Accuracy: 0.8122
Epoch 75/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.1750 Accuracy: 0.9349
100%|██████████| 14/14 [00:04<00:00,  3.29it/s]
test Loss: 0.5914 Accuracy: 0.7896
Epoch 76/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.2015 Accuracy: 0.9241
100%|██████████| 14/14 [00:04<00:00,  3.33it/s]
test Loss: 0.8854 Accuracy: 0.7738
Epoch 77/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.2167 Accuracy: 0.9128
100%|██████████| 14/14 [00:03<00:00,  3.61it/s]
test Loss: 0.4440 Accuracy: 0.7896
Epoch 78/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.2052 Accuracy: 0.9275
100%|██████████| 14/14 [00:03<00:00,  3.85it/s]
test Loss: 0.2603 Accuracy: 0.8665
Epoch 79/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.1835 Accuracy: 0.9264
100%|██████████| 14/14 [00:03<00:00,  3.84it/s]
test Loss: 0.4715 Accuracy: 0.8100
Epoch 80/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.1954 Accuracy: 0.9213
100%|██████████| 14/14 [00:03<00:00,  3.77it/s]
test Loss: 0.4479 Accuracy: 0.8054
Epoch 81/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.1932 Accuracy: 0.9241
100%|██████████| 14/14 [00:03<00:00,  3.83it/s]
test Loss: 0.3193 Accuracy: 0.8394
Epoch 82/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.1761 Accuracy: 0.9337
100%|██████████| 14/14 [00:03<00:00,  3.86it/s]
test Loss: 0.3013 Accuracy: 0.8552
Epoch 83/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.1827 Accuracy: 0.9264
100%|██████████| 14/14 [00:03<00:00,  3.79it/s]
test Loss: 0.2696 Accuracy: 0.8756
Epoch 84/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.1731 Accuracy: 0.9349
100%|██████████| 14/14 [00:03<00:00,  3.55it/s]
test Loss: 0.5238 Accuracy: 0.8235
Epoch 85/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.1861 Accuracy: 0.9326
100%|██████████| 14/14 [00:04<00:00,  3.33it/s]
test Loss: 0.4157 Accuracy: 0.8145
Epoch 86/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.2179 Accuracy: 0.9162
100%|██████████| 14/14 [00:04<00:00,  3.36it/s]
test Loss: 0.3809 Accuracy: 0.8303
Epoch 87/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.1806 Accuracy: 0.9309
100%|██████████| 14/14 [00:03<00:00,  3.66it/s]
test Loss: 0.2551 Accuracy: 0.8891
Epoch 88/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.22it/s]
train Loss: 0.2328 Accuracy: 0.9043
100%|██████████| 14/14 [00:03<00:00,  3.88it/s]
test Loss: 0.3001 Accuracy: 0.8891
```

```
Epoch 89/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.24it/s]
train Loss: 0.2293 Accuracy: 0.9134
100%|██████████| 14/14 [00:03<00:00,  3.82it/s]
test Loss: 0.5079 Accuracy: 0.7919
Epoch 90/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.1972 Accuracy: 0.9173
100%|██████████| 14/14 [00:03<00:00,  3.89it/s]
test Loss: 0.4524 Accuracy: 0.8303
Epoch 91/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.1840 Accuracy: 0.9326
100%|██████████| 14/14 [00:03<00:00,  3.79it/s]
test Loss: 0.2479 Accuracy: 0.8937
Epoch 92/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.1862 Accuracy: 0.9349
100%|██████████| 14/14 [00:03<00:00,  3.81it/s]
test Loss: 0.6861 Accuracy: 0.8190
Epoch 93/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.1797 Accuracy: 0.9332
100%|██████████| 14/14 [00:04<00:00,  3.47it/s]
test Loss: 0.2162 Accuracy: 0.9005
Epoch 94/100
----------
100%|██████████| 56/56 [00:24<00:00,  2.25it/s]
train Loss: 0.1554 Accuracy: 0.9360
100%|██████████| 14/14 [00:04<00:00,  3.25it/s]
test Loss: 0.2293 Accuracy: 0.9095
Epoch 95/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.2109 Accuracy: 0.9122
100%|██████████| 14/14 [00:03<00:00,  3.58it/s]
test Loss: 0.9750 Accuracy: 0.7240
Epoch 96/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.1645 Accuracy: 0.9394
100%|██████████| 14/14 [00:03<00:00,  3.83it/s]
test Loss: 0.2565 Accuracy: 0.8778

Epoch 97/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.1712 Accuracy: 0.9332
100%|██████████| 14/14 [00:03<00:00,  3.89it/s]
test Loss: 0.4952 Accuracy: 0.8077
Epoch 98/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.1767 Accuracy: 0.9304
100%|██████████| 14/14 [00:03<00:00,  3.85it/s]
test Loss: 0.2226 Accuracy: 0.9140
Epoch 99/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.23it/s]
train Loss: 0.1804 Accuracy: 0.9281
100%|██████████| 14/14 [00:03<00:00,  3.86it/s]
test Loss: 0.4323 Accuracy: 0.8416
Epoch 100/100
----------
100%|██████████| 56/56 [00:25<00:00,  2.24it/s]
train Loss: 0.1980 Accuracy: 0.9304
100%|██████████| 14/14 [00:03<00:00,  3.88it/s]test Loss: 0.3034 Accuracy: 0.8167
Training completed in 48m 9s
Best test accuracy: 0.9140
```

Anexo 4: Fichas de Registro del proceso de datos.

Ficha de Registro nro.1.- Características de los videos capturados en campo

ITEM	AVENIDAS DE ESTUDIO DE CAPTURA DE VIDEOS	DURACIÓN DE VIDEO (min. y seg.)	VELOCIDAD DE FOTOGRAMA (fotogramas/seg.)	ANCHO DE FOTOGRAMA (píxeles)	ALTO DE FOTOGRAMA (píxeles)
1.00	Av. Miguel Grau (Ida)	20:24	30.02	1080	1920
2.00	Av. Miguel Grau (Vuelta)	19:08	24.02	1080	1920
3.00	Av. Don Bosco (Ida)	11:05	30.02	1080	1920
4.00	Av. Don Bosco (Vuelta)	12:11	26.25	1080	1920
5.00	Av. Andrés Avelino Cáceres (Inicio)	02:14	30.03	1080	1920
6.00	Av. Andrés Avelino Cáceres (Ida)	10:54	30.02	1080	1920
7.00	Av. Andrés Avelino Cáceres (Vuelta)	12:38	25.98	1080	1920
8.00	Av. Cesar Vallejo (Ida y Vuelta)	04:56	24.00	1080	1920
9.00	Av. Sánchez Cerro (Ida)	03:37	24.01	1080	1920

Nota. Toda información se ha validado en coordinación con el DR. Alan Fischer Ayala Obregón.

Ficha de Registro nro.2.- Cantidad de imágenes extraídas de los videos de campo

ITEM	AVENIDAS DE ESTUDIO DE CAPTURA DE VIDEOS	CANTIDAD DE IMÁGENES EXTRAÍDAS DE VIDEOS
1.00	Av. Miguel Grau (Ida)	36737
2.00	Av. Miguel Grau (Vuelta)	28919
3.00	Av. Don Bosco (Ida)	21321
4.00	Av. Don Bosco (Vuelta)	19207
5.00	Av. Andrés Avelino Cáceres (Inicio)	4048
6.00	Av. Andrés Avelino Cáceres (Ida)	19646
7.00	Av. Andrés Avelino Cáceres (Vuelta)	19707
8.00	Av. Cesar Vallejo (Ida y Vuelta)	7121
9.00	Av. Sánchez Cerro (Ida)	5213
TOTAL		161919

Nota. Toda información se ha validado en coordinación con el DR. Alan Fischer Ayala Obregón.

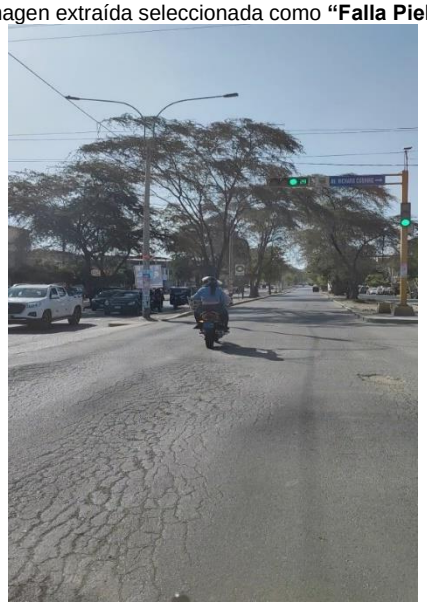
En la Ficha de Registro nro.3, las dimensiones de las imágenes son de 1080 píxeles de ancho por 1920 píxeles de alto, tal como se ilustra en la Figura nro. 1 adjunta.

Ficha de Registro nro.3.- Cantidad de imágenes seleccionadas de acuerdo con su tipo de falla.

ITEM	AVENIDAS DE ESTUDIO DE CAPTURA DE VIDEOS	CANTIDAD DE IMÁGENES EXTRAÍDAS DE VIDEOS	FALLA FISURA LONGITUDINAL	FALLA TRANSVERSAL	FALLA PIEL DE COCODRILO
1.00	Av. Miguel Grau (Ida)	36737	66	109	57
2.00	Av. Miguel Grau (Vuelta)	28919	51	119	70
3.00	Av. Don Bosco (Ida)	21321	37	30	30
4.00	Av. Don Bosco (Vuelta)	19207	47	30	43
5.00	Av. Andrés Avelino Cáceres (Inicio)	4048	0	0	0
6.00	Av. Andrés Avelino Cáceres (Ida)	19646	38	29	28
7.00	Av. Andrés Avelino Cáceres (Vuelta)	19707	8	12	58
8.00	Av. Cesar Vallejo (Ida y Vuelta)	7121	19	7	15
9.00	Av. Sánchez Cerro (Ida)	5213	38	12	6
TOTAL		**161919**	**304**	**348**	**307**

Nota. Toda información se ha validado en coordinación con el DR. Alan Fischer Ayala Obregón.

Figura nro. 1.-Imagen extraída seleccionada como **"Falla Piel de cocodrilo".**



En la Ficha de Registro nro.4, las dimensiones de las imágenes se establecen de acuerdo con el primer recorte de 800 píxeles de ancho por 450 píxeles de alto, tal como se ilustra en la Figura nro. 2.

Ficha de Registro nro.4.- Cantidad de imágenes obtenidas a través del primer Recorte.

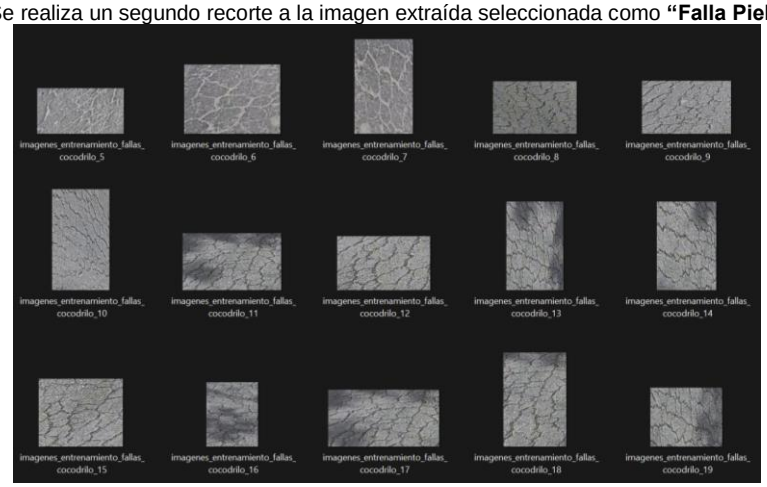
ITEM	AVENIDAS DE ESTUDIO DE CAPTURA DE VIDEOS	CANTIDAD DE IMÁGENES SELECCIONADAS EN FICHA DE REGISTRO N°3	IMÁGENES FALLA FISURA LONGITUDINAL	IMÁGENES FALLA TRANSVERSAL	IMÁGENES FALLA PIEL DE COCODRILO
1.00	Av. Miguel Grau (Ida)	232	60	116	35
2.00	Av. Miguel Grau (Vuelta)	240	50	143	75
3.00	Av. Don Bosco (Ida)	97	36	36	30
4.00	Av. Don Bosco (Vuelta)	120	64	30	49
5.00	Av. Andrés Avelino Cáceres (Inicio)	0	0	0	0
6.00	Av. Andrés Avelino Cáceres (Ida)	95	35	34	28
7.00	Av. Andrés Avelino Cáceres (Vuelta)	78	8	13	58
8.00	Av. Cesar Vallejo (Ida y Vuelta)	41	19	7	22
9.00	Av. Sánchez Cerro (Ida)	56	38	12	7
TOTAL		**959**	**310**	**391**	**304**

Nota. Toda información se ha validado en coordinación con el DR. Alan Fischer Ayala Obregón.

Figura nro. 2.- Se recorta la imagen extraída seleccionada como **"Falla Piel de cocodrilo".**

Aquí, las dimensiones de las imágenes se establecen de acuerdo con el segundo recorte selectivo, teniendo en cuenta un criterio importante: "**el algoritmo no debe extraer demasiada información, por lo que es necesario extraer solo la falla**", los recortes se ilustran en la figura nro. 3 y la cantidad obtenida en la Ficha de Registro nro.5.

Figura nro. 3.- Se realiza un segundo recorte a la imagen extraída seleccionada como **"Falla Piel de cocodrilo".**



Ficha de Registro nro.5.- Cantidad de imágenes obtenidas a través del segundo recorte y aplicación del proceso de aumento del conjunto de datos.

Tipo de falla	Cantidad de imágenes
Longitudinales	1104
Transversal	1104
Piel cocodrilo	1104
Total	**3312**

Nota. Toda información se ha validado en coordinación con el DR. Alan Fischer Ayala Obregón.

Ficha de Registro nro.6.- Se asigna el porcentaje de la cantidad de imágenes que se coloca en las carpetas de nombre "train" y "test" para proceder al entrenamiento.

Tipo de falla	Carpeta asignada	Cantidad en % que se le debe asignar a cada tipo de fallas	Cantidad de imágenes por su porcentaje asignada	Total, de imágenes
Longitudinales	train	80	883	1104
	test	20	221	
Transversal	train	80	883	1104
	test	20	221	
Piel cocodrilo	train	80	883	1104
	test	20	221	

Nota. Toda información se ha validado en coordinación con el DR. Alan Fischer Ayala Obregón.

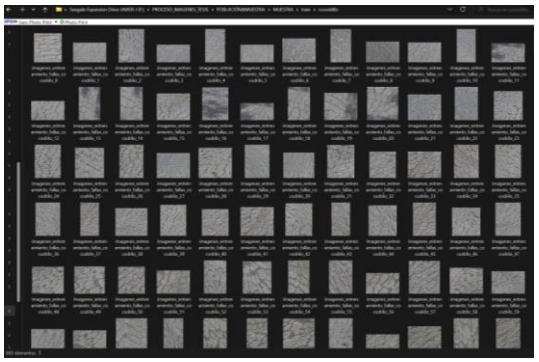


Figura nro.6.- Se guardan en la carpeta "train" 883 imágenes del tipo de "falla piel de cocodrilo".

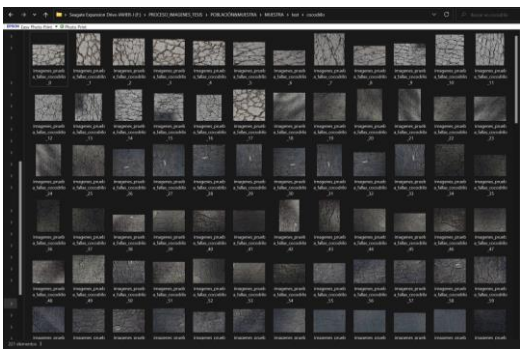
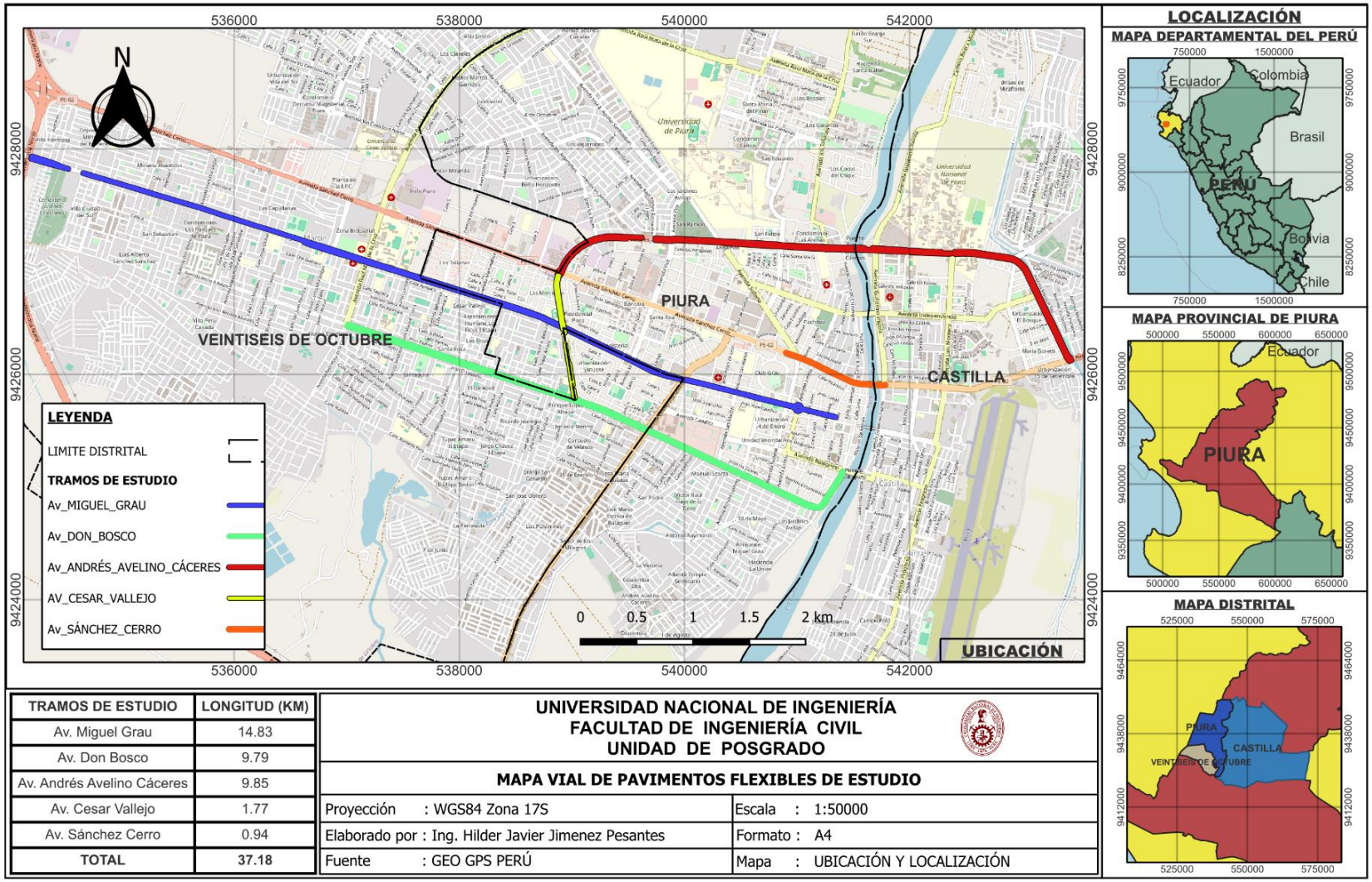


Figura nro.5.- Se guardan en la carpeta "test" 221 imágenes del tipo de "falla piel de cocodrilo".

Anexo 5: Mapa vial de los pavimentos flexibles recorridos.



TRAMOS DE ESTUDIO	LONGITUD (KM)
Av. Miguel Grau	14.83
Av. Don Bosco	9.79
Av. Andrés Avelino Cáceres	9.85
Av. Cesar Vallejo	1.77
Av. Sánchez Cerro	0.94
TOTAL	37.18

UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE INGENIERÍA CIVIL
UNIDAD DE POSGRADO

MAPA VIAL DE PAVIMENTOS FLEXIBLES DE ESTUDIO

Proyección : WGS84 Zona 17S	Escala : 1:50000
Elaborado por : Ing. Hilder Javier Jimenez Pesantes	Formato : A4
Fuente : GEO GPS PERÚ	Mapa : UBICACIÓN Y LOCALIZACIÓN