UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Detección de defectos estructurales en saneamiento usando redes neuronales

Informe de Proyecto de Grado presentado por

Santiago Acquarone, Santiago Costa y Federico Dallo

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Gonzalo Tejera
Mercedes Marzoa

Montevideo, 7 de septiembre de 2024

Agradecimientos

A nuestras familias, amigos, supervisores y a todos los que nos han acompañado en este camino.

Resumen

Los sistemas de saneamiento son fundamentales para la infraestructura urbana, pero con el tiempo, las tuberías experimentan un deterioro que afecta su funcionamiento. Por ello, es crucial realizar inspecciones rutinarias. La limitada disponibilidad de profesionales frente al considerable número de tuberías hace imperativa la automatización de las inspecciones.

En este trabajo se presenta una investigación sobre el uso de redes neuronales para afrontar el problema de la detección de defectos en tuberías de saneamiento a través de imágenes. Se busca aportar al área de la automatización de las inspecciones de saneamiento, creando modelos que puedan procesar video-inspecciones y detectar defectos.

El enfoque principal se centra en entrenar modelos de clasificación de imágenes usando el conjunto de datos Sewer-ML. Este conjunto dispone más de un millón de imágenes de tuberías de saneamiento clasificadas por profesionales del rubro.

Se exploran distintas familias de modelos de clasificación de imágenes. Entre ellos *Inception*, *ResNet*, *DenseNet*, *ViT* y *DaViT*. Se evalúan los modelos generados en términos de características y métricas de desempeño.

Se presenta a su vez la metodología utilizada para experimentar con estos modelos y los programas utilizados. Para el manejo de redes neuronales se utiliza principalmente el lenguaje Python y la biblioteca PyTorch. Para la infraestructura se utilizó la plataforma ClusterUY.

Por último, se implementa en Python un sistema que facilita el uso de las redes generadas para una empresa local con el fin de automatizar la generación de reportes a partir de video-inspecciones.

Palabras clave: Redes neuronales, Saneamiento, Aprendizaje profundo, Visión computacional, Clasificación de imágenes, Detección de defectos, Inception, ResNet, DenseNet, ViT, DaViT

Índice general

Capítulo 1

Introducción

1.1. Presentación del problema

Los sistemas de tuberías de saneamiento poseen un papel primordial en la infraestructura urbana al ser los responsables de transportar las aguas residuales de una ciudad a los distintos puntos de desecho. Con su uso prolongado las tuberías están sometidas a un deterioro estructural continuo que reduce la capacidad de su funcionamiento, provocando la ocurrencia de varios defectos que potencialmente conducen a daños estructurales severos. En consecuencia, es imprescindible la puesta en marcha de rutinas de inspección de tuberías de forma regular, con el propósito de detectar defectos en una etapa temprana y tomar las medidas apropiadas para lidiar con los mismos, consiguiendo así reducir y mitigar el ritmo de deterioro de estos sistemas.

El método tradicional de detección de defectos radica en la inspección visual realizada por profesionales en el área. Sin embargo, estas tareas son muy laboriosas y demandantes de tiempo, especialmente al momento de realizar el control y monitoreo simultáneo de varias locaciones defectuosas, teniendo en cuenta que los sistemas de saneamiento consisten de redes complejas de tuberías que cubren miles de kilómetros.

Adicionalmente, estas tareas son realizadas con menor frecuencia dadas las condiciones sanitarias adversas que presentan estos entornos y la propensión a accidentes en ellos, además del costo económico que puede requerir el despliegue de equipamiento y de personal al momento de realizar inspecciones. Consiguientemente, se han adoptado distintos enfoques que tienden a la automatización de las tareas de inspección de tuberías mediante el empleo de tecnología.

Diversas investigaciones han revelado información acerca de métodos que buscan automatizar estas tareas de detección. Uno de ellos y ampliamente utilizado, es la inspección mediante tecnología CCTV (tecnología de videovigilancia "televisión de circuito cerrado", del inglés Closed Circuit Television). Se utilizan dispositivos robóticos que son introducidos en el interior de las tuberías y que son operados por un técnico desde la superficie. Estos dispositivos cuentan con

cámaras para la recolección de contenido fotográfico o audiovisual para luego ser evaluado por profesionales y tomar las medidas pertinentes. Si bien el uso de CCTV mejora la seguridad al no requerir que una persona ingrese en la tubería, aún se requiere de personal técnico para la inspección detallada de videos, hecho que entraña una labor intensiva y tediosa con una demanda de tiempo sustancial. Esto sumado a la cantidad limitada de profesionales y el significante número de sistemas de saneamientos, dificultan la viabilidad y escalabilidad de este método. Urge entonces la necesidad de automatizar este proceso con el fin de minimizar la intervención humana y disminuir los tiempos de detección. En este sentido, los métodos de detección automatizada basada en visión artificial y técnicas de aprendizaje profundo han tenido un fuerte impacto en el campo de la detección de defectos en sistemas de saneamiento.

Este proyecto de grado surge en relación a una línea de investigación del grupo MINA perteneciente al Instituto de Computación de Facultad de Ingeniería. Esta se desarrolla en el marco del consorcio formado por la Fundación Julio Ricaldoni y las empresas DICA & Asociados y Kreitech. En este contexto los aspectos de robótica autónoma son liderados por el grupo MINA.

Como parte del trabajo en automatización de las tareas de inspección que desarrolla el grupo, se encuentra la tarea de procesar los videos recolectados por los robots y procesarlos para ubicar los defectos en un modelo virtual de la tubería de saneamiento usando odometría. En este sentido, surge el interés en el desarrollo de redes neuronales para la detección de defectos en saneamiento.

Por otra parte, el grupo MINA colabora con la empresa DICA & Asociados. Esta empresa brinda servicios de ingeniería varios, entre ellos, inspecciones de tuberías de saneamiento, las cuales realizan empleando robots con tecnología CCTV para clientes como la Intendencia de Montevideo.

En este proyecto, se investigó el uso de redes neuronales para la detección de defectos estructurales en saneamiento a partir de un video, con el fin de aportar al trabajo que realiza el grupo MINA. Como parte de este trabajo, se entrenó y evaluó distintos modelos para la tarea.

Además, se construyó una solución para la utilización de estas redes neuronales por parte de la empresa DICA & Asociados, con el fin de automatizar su proceso de trabajo.

1.2. Estructura del documento

El presente documento se estructura en distintas secciones, comenzando por el marco teórico donde se exhiben conceptos introductorios que sustentan el trabajo en cuestión, sucedido por una sección dedicada a la revisión de antecedentes en la que se presentan trabajos anteriores relacionados.

Posteriormente, se encuentra el desarrollo del proyecto donde se expone el trabajo elaborado abarcando las distintas etapas, los desafíos afrontados y las decisiones y estrategias tomadas para abordarlos.

Por último, se encuentra la sección de conclusiones y trabajo futuro.

Capítulo 2

Marco teórico

En este capítulo se presentan y definen conceptos teóricos necesarios para una mayor comprensión del trabajo realizado. Particularmente, se pretende introducir al lector acerca de los defectos en saneamiento, sus categorías y tipos, seguido de una introducción a los problemas de clasificación y detección, finalizando con una presentación sobre redes neuronales y el concepto de *fine-tuning*.

2.1. Defectos en saneamiento

Existe una organización llamada NASSCO (National Association of Sewer Service Companies, (Nassco, s.f.)) que tiene por objeto establecer estándares para el diagnóstico, mantenimiento y rehabilitación de infraestructura subterránea. Esta misma ha desarrollado el Programa de Certificación para la Evaluación de Tuberías (del inglés *Pipeline Assessment and Certification Program*, PACP) (Hanley y Drinkwater, 2020) cuyo propósito es proveer estandarización y consistencia en la tarea de evaluación de las condiciones de la infraestructura subterránea. En este sentido, se especifica un método estándar que permite codificar los distintos defectos de manera consistente y confiable, diagnosticar la condición de los activos y poder así planificar el mantenimiento, la rehabilitación o el reemplazo de tuberías, adecuadamente.

El PACP, asimismo, divide a los defectos en dos grandes categorías, estructurales, y de operación y mantenimiento.

Defectos Estructurales

El programa describe distintas familias o grupos de defectos estructurales junto con una codificación asociada, indicando el daño o deficiencia en la tubería. A continuación se describen dichos grupos.

- Grieta (C, del inglés *Crack*)
- Fractura (F, del inglés *Fracture*)

- Rotura (B, del inglés *Broken*)
- Agujero (H, del inglés *Hole*)
- Deformación (D, del inglés *Deformed*)
- Colapso (X, en inglés es *Collapse*, que al estar la C ya usada utilizan la X)
- Junta (J, del inglés *Joint*)
- Daño Superficial (S, del inglés *Surface Damage*)
- Características del Revestimiento (LF, del inglés *Lining Features*)
- Falla de la Soldadura (WF, del inglés *Weld Failure*)
- Reparación Puntual (RP, del inglés *Point Repair*)
- Mampostería (del inglés *Brickwork*, no tiene código)

En la figura 2.1 se presentan ejemplos de algunos de los defectos mencionados

Según sea el material de construcción de la tubería, se frecuentarán ciertos defectos más que otros. Por ejemplo, en caso de materiales rígidos como lo son el concreto o ladrillo, exhiben una mayor proporción de grietas y fracturas. Si no se toman medidas preventivas, estos podrían sufrir un mayor deterioro produciendo roturas, agujeros y deformaciones que comprometan la integridad estructural de la tubería. Grietas o fracturas pequeñas no necesariamente denotan una situación de falla estructural. Asimismo, materiales flexibles como el metal corrugado o el plástico responden de maneras distintas frente a diversas condiciones de carga. Estos, se deforman mientras continúan resistiendo a cargas externas previo a la formación de grietas y fracturas. Aún así, son susceptibles a estas últimas e incluso a eventuales colapsos en caso de descuidos en su mantenimiento. La magnitud de deformación tolerable de una tubería esta estrechamente ligada a su material de construcción.

A su vez, dentro de estas familias mencionadas, existen distintos tipos de defectos de mayor especificidad que, a su vez, reciben una codificación asociada.

Defectos de Operación y Mantenimiento (O&M)

Análogamente, el PACP presenta y define una familia de códigos donde se describen distintos tipos de objetos anómalos que son hallados en las tuberías, pudiendo interferir en el correcto funcionamiento de las mismas. Estos objetos se comprenden dentro de las siguientes familias.

- Depósitos - Incluye tres subgrupos: Adheridos, Asentados e Incorporados (DA, del inglés *Deposits Attached*/DS, del inglés *Deposits Settled*/DN, del inglés *Deposits Ingress*)
- Raíces (R, del inglés *Roots*) (ejemplo en la figura 2.2)

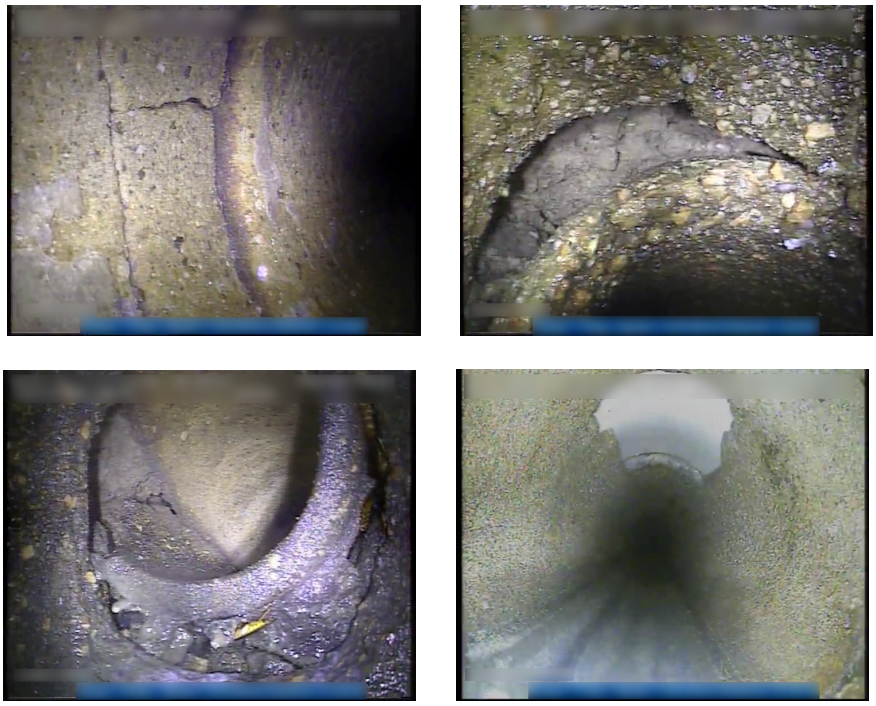


Figura 2.1: Ejemplos de defectos estructurales de fractura (esquina superior izquierda), desfasaje de junta (esquina superior derecha), desfasaje de junta y rotura (esquina inferior izquierda), y reparación puntual (esquina inferior derecha). Figuras provistas por DICA & Asociados.

- Infiltración (I, del inglés *Infiltration*)
- Obstáculos/Obstrucciones (OB, del inglés *Obstacles/Obstructions*)
- Alimañas (V, del inglés *Vermin*)
- Pruebas y Lechadas (G, del inglés *Grout Test & Seal*)

Estos tipos de defectos pueden aumentar significativamente el grado y velocidad de deterioro de los sistemas de saneamiento si no se efectúa un mantenimiento rutinario de carácter preventivo, acortándose así, su vida útil.

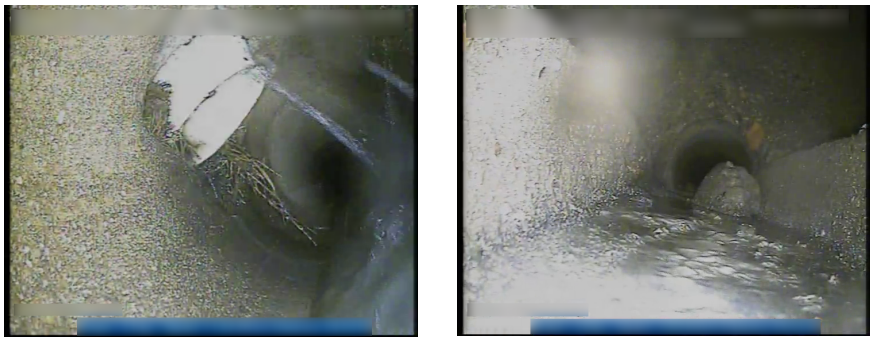


Figura 2.2: Ejemplos de defectos de operación y mantenimiento: raíces (izquierda) y obstáculos (derecha). Figuras provistas por DICA & Asociados.

2.2. Clasificación de imágenes y detección de objetos

La clasificación de imágenes y la detección de objetos son dos de los problemas principales del campo de la visión computacional. Comprender la diferencia entre estas dos es fundamental para una decisión acertada en el acercamiento a utilizar en relación a un propósito dado.

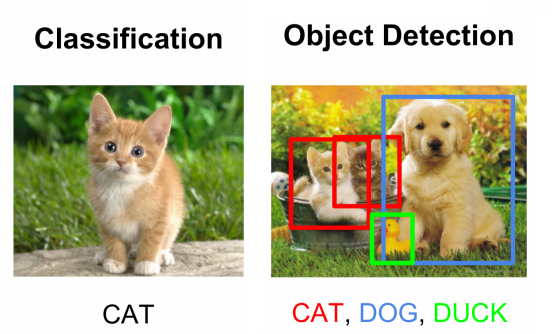


Figura 2.3: Ejemplo de la tarea de clasificación y detección de objetos, respectivamente. Figura extraída de (Bhatia, 2017).

La **clasificación de imágenes** consiste en la asignación de etiquetas o clases a una cierta imagen basada en su contenido. Estas etiquetas o clases pertenecen a un conjunto predefinido que, por lo general, están asociadas a un dominio específico. Existen distintas técnicas que permiten la extracción de información o características de imágenes con el fin de realizar predicciones en la asignación de clases a una cierta imagen. Una de ellas y asociada al campo del aprendizaje profundo, son las redes neuronales convolucionales que han probado obtener

resultados notables y ser el estado del arte.

Por otra parte, la tarea de **detección de objetos** alude al proceso de localizar objetos de interés dentro de una imagen y clasificarlos. Para ello, se requiere identificar el objeto en la imagen proveyendo cuadros delimitadores (*bounding boxes*) que encierren lo más estrechamente posible al objeto detectado (ver figura 2.3). Los métodos o algoritmos empleados examinan diferentes regiones o áreas de la imagen para la identificación de potenciales objetos. La mayoría de estos, asociados al campo del aprendizaje profundo (Bashar, 2023).

2.2.1. Problemas de clasificación

Existen distintos problemas inherentes a la tarea de clasificación dentro de los cuales se destacan tres muy populares, la clasificación binaria (*binary*), multi-clase (*multi-class*) y multi-etiqueta (*multi-label*).

Se denomina **clasificación binaria** a la tarea de asignar a cierta instancia o dato una y solo una etiqueta o clase de un conjunto de dos posibles que son mutuamente excluyentes. Aplicaciones de uso muy comunes son aquellas de carácter si/no, positivo/negativo tales como la detección de *spam* o de fraude (pirzada, 2023).

Por otra parte, en la **clasificación multi-clase**, cada dato es asignado una etiqueta perteneciente a un conjunto de más de dos clases predefinidas. A diferencia de la clasificación binaria, la idea es distinguir una instancia entre múltiples clases o etiquetas y asignarle la más probable. Escenarios de uso comunes aluden a casos donde la esencia se encuentra en la categorización de elementos, como por ejemplo análisis de sentimiento, clasificación de textos, entre otros (GeeksforGeeks, 2024).

Finalmente, la **clasificación multi-etiqueta** guarda notable similitud con la anterior, con la distinción de que a una instancia o dato se le puede designar múltiples etiquetas simultáneamente. Mientras que en la anterior una instancia recibe una única etiqueta exclusiva, la clasificación multi-etiqueta amplía el universo permitiendo que instancias exhiban características comprendidas en varias categorías reflejando, asimismo, una mayor complejidad en la tarea. Casos de usos comunes son la clasificación de imágenes con múltiples etiquetas o etiquetado de documentos donde estos se encuentran incluidos en varias categorías (GeeksforGeeks, 2024).

En la figura 2.4 se observa un ejemplo para cada problema de clasificación, donde la o las etiquetas más probables (según el caso) se marcan en verde.

Habitualmente, estos problemas de clasificación se suelen abordar mediante el entrenamiento de métodos o modelos que toman como insumo grandes volúmenes de datos etiquetados según los requerimientos del procedimiento del

acercamiento a seguir. Consiguientemente, se emplean estos modelos sobre datos desconocidos (no etiquetados) pertenecientes también al mismo dominio, de manera de clasificarlos y asignarles las etiquetas más probables.

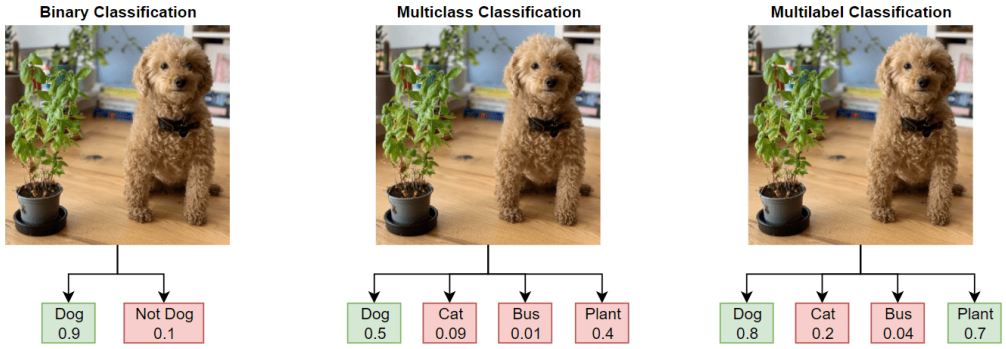


Figura 2.4: Ejemplos clasificación binaria, multi-clase y multi-etiqueta, respectivamente. Figura extraída de (MathWorks, s.f.).

2.3. Redes neuronales

Las redes neuronales son un modelo computacional inspirado en el funcionamiento de las neuronas biológicas. Estos modelos son ampliamente utilizados en el área del aprendizaje automático debido a su capacidad para resolver problemas de alta complejidad en cuanto al procesado de información. Estas redes son capaces de aprender a reconocer diversos patrones y relaciones en los datos a través de sus conexiones internas.

2.3.1. Estructura

Una red neuronal esta compuesta por nodos, comúnmente denominados neuronas artificiales, que realizan operaciones matemáticas en base a ciertos valores de entradas, produciendo como salida un nuevo valor. A su vez, estos nodos están agrupados en distintas capas pudiendo tener cada uno conexiones hacia nodos de una capa anterior o posterior. Se disciernen tres tipos de capas en una red, siendo la primera una capa de entrada que recibe datos a procesar, una de salida que devuelve los resultados del procesamiento de dichos datos, y por último, varias capas ocultas situadas entre estas últimas dos. Las conexiones entre nodos son direccionales sugiriendo que la información fluye en un solo sentido a través de la red; desde la capa de entrada hacia la de salida (ver figura 2.5). Mientras las capas de entrada y salida son siempre una, las ocultas pueden variar en su cantidad e inclusive ser nulas. El número de nodos (en cada capa) y de capas ocultas dependerá de la tarea que se desee realizar (IBM, s.f.-b).

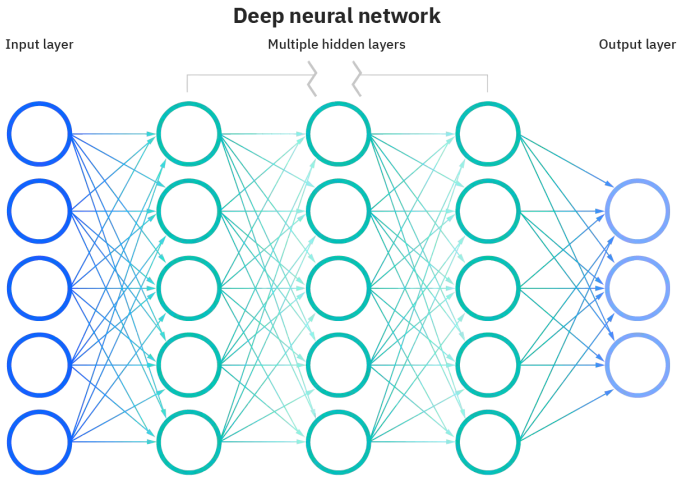


Figura 2.5: Ejemplo de una red neuronal con cinco nodos en la capa de entrada (*input layer*), tres capas ocultas (*hidden layers*), y una capa de salida (*output layer* con tres nodos. Figura extraída de (Umer, 2022).

2.3.2. Funcionamiento general

Cada nodo tiene asociado un conjunto de pesos $w_1, ..., w_n$ y un sesgo b. Los valores de entrada $x_1, ..., x_n$ que el nodo recibe son multiplicados por su respectivo peso w_i, y luego sumados junto con el sesgo. Al valor obtenido se le aplica una función de activación f (ver ecuación 2.1) cuyo resultado corresponderá a la salida de la neurona que, a su vez, se convierte en un dato de entrada para los nodos de la siguiente capa.

$$f\left(\sum_{i=1}^{n} x_i w_i + b\right) \tag{2.1}$$

Los pesos designados de un nodo determinan la importancia o atención dada a los datos de la entrada, donde aquellos más grandes contribuyen de manera más significativa en el valor de salida.

Inicialmente, estos pesos reciben valores aleatorios por lo cual es previsible que la salida de la red no coincida con la esperada. Es aquí donde surge el concepto de "aprendizaje" de una red, proceso que radica en la tarea de ajustar dichos pesos de cada nodo de modo que la salida de la red se adecue a la esperada, o en otras palabras, disminuir el error entre la salida predicha por la red y la esperada o real. Para ello, se define una función de pérdida que permite estimar la diferencia entre las predicciones efectuadas por la red y los datos o valores reales de entrenamiento. Esta función es optimizada por un algoritmo denominado *backpropagation* que ajusta el valor de los pesos estimando el efecto

que cada uno tiene sobre el error total de la salida de la red, minimizando así la función de pérdida susodicha.

2.4. Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN, del inglés *Convolutional Neural Network*) son redes neuronales ampliamente utilizadas para el reconocimiento y clasificación de imágenes.

Previo a estas redes, se empleaban métodos manuales de extracción de características para la identificación de objetos en imágenes, los cuales demandaban un tiempo considerable. No obstante, las redes convolucionales proveen un enfoque más escalable a las tareas de clasificación de imágenes y reconocimiento de objetos, capitalizando ciertos principios matemáticos para la identificación de patrones dentro de una imagen.

2.4.1. Estructura

En la arquitectura de una red convolucional se distinguen, primordialmente, tres tipos de capas; capa convolucional (*convolutional layer*), capa de agrupación (*pooling layer*) y capa densamente conectada (*Fully-connected layer*).

La primer capa de la red es convolucional que puede ser sucedida por varias del mismo tipo o de agrupamiento, mientras que la final consiste en una única capa densamente conectada. Con cada capa, la red incremente su complejidad permitiendo una identificación más precisa de distintas áreas de la imagen. Las capas iniciales se enfocan en características simples como colores y aristas. A medida que los datos de la imagen progresan a través de las capas, se reconocen elementos de mayor tamaño o formas del objeto hasta lograr distinguirlo completamente.

2.4.2. Funcionamiento general

Con la finalidad de obtener una visión integral del funcionamiento de una red convolucional, primeramente se debe ahondar en las tareas que realiza cada una de las capas.

Las **capas convolucionales** son el bloque constructivo esencial de estas redes y donde ocurre la mayor parte del cómputo.

Estas capas reciben como entrada una imagen, precisamente, su codificación como una matriz de píxeles. A modo de ejemplo, si la entrada consiste en una imagen a color, su matriz respectiva tendrá tres dimensiones donde cada una corresponde a un canal de color correspondiente al código RGB (rojo, verde y azul, del inglés *Red, Green, Blue*). Adicionalmente, cada capa posee una cierta cantidad de detectores de patrones denominados filtros (*filters*); cada uno se interpreta como una matriz pequeña de pesos de dos dimensiones de tamaño variable que, típicamente, suele ser de 3x3 y detectan un patrón en particular.

Cada filtro se desplaza sucesivamente sobre la imagen de izquierda a derecha y de arriba hacia abajo, y en cada área de la imagen aplicado se efectúa una operación matemática entre los píxeles de dicha área y los valores del filtro, conocida como convolución. El resultado de esta serie de operaciones convolucionales entre la imagen de entrada y el filtro se lo conoce como mapa de características (del inglés, *feature map*), en definitiva una matriz también, que devuelve como salida la red. Consecuentemente, la cantidad de filtros utilizados determinará la cantidad de mapas de salida (IBM, s.f.-a).

A modo de ilustración, en la figura 2.6 se aprecia como el filtro (en naranja) se desplaza sucesivamente sobre la imagen de entrada (en azul) realizando una convolución para cada área de 3x3, hasta abarcar su totalidad, almacenando cada resultado de dicha operación en un mapa de característica (en verde) que se devuelve como salida.

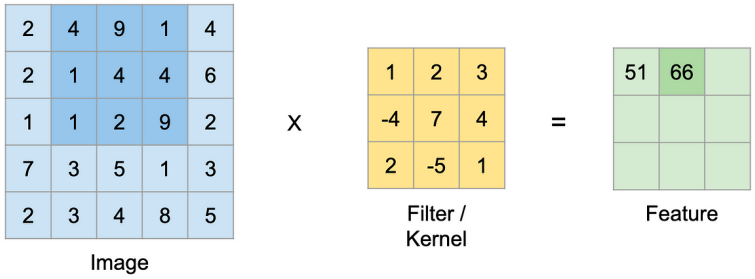


Figura 2.6: Ejemplo de aplicación de un filtro (en naranja) de 3x3 sobre una imagen de entrada (azul) devolviendo como resultado un mapa de características (verde). Figura extraída de (Patel, 2019).

Asimismo, los mapas de salida de una capa convolucional pueden comprender grandes cantidades de parámetros, especialmente si la imagen de entrada es de tamaño considerable, dificultando la eficiencia en el manejo de los datos conforme se incrementa la profundidad de la red.

Surgen entonces, las **capas de agrupamiento**, responsables de la reducción en la dimensionalidad de los parámetros de dichos mapas. Esto permite aminorar los recursos de cómputos requeridos para el procesamiento de los datos y, a la vez, la extracción de características o patrones dominantes de la imagen. Existen distintas técnicas de agrupamiento, una de ellas, *Max Pooling*, que similarmente al proceso de convolución, existe un filtro que se desplaza sobre la imagen, con la distinción de que en cada área de la imagen donde es aplicado, se extrae el valor más grande (Saha, 2018). Esto permite reducir la dimensionalidad manteniendo información relevante a la vez.

En la figura 2.7 se visualiza un ejemplo ilustrativo de *Max Pooling* empleando un filtro de 2x2 sobre una entrada de 4x4, reduciendo su dimensionalidad a una

salida de 2x2.

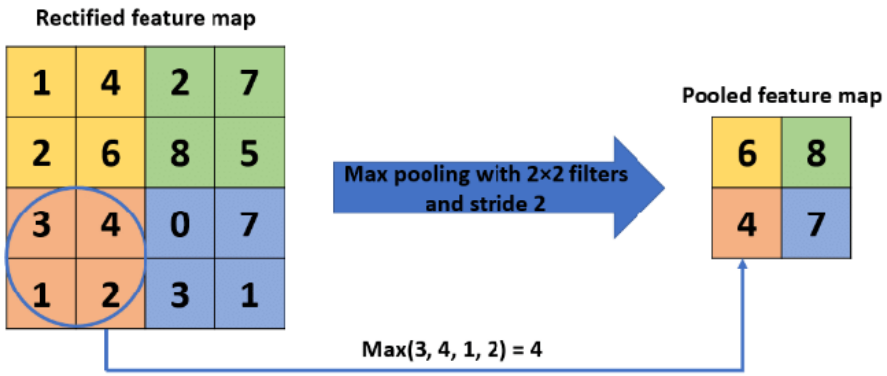


Figura 2.7: Ejemplo *Max Pooling* utilizando un filtro de 2x2 sobre una entrada de 4x4. Figura extraída de (Gholamalinejad, 2020).

Usualmente, se dispone una capa de agrupamiento luego de una convolucional, de manera de reducir la dimensión de los datos a medida que se incrementa el número de capas, y por ende, la complejidad de la red (ver figura 2.8). Si bien una red más profunda posibilita la extracción de patrones más sofisticados, incrementa asimismo los recursos de cómputo requeridos.

Finalmente, la capa **densamente conectada** consiste en una red neuronal donde cada nodo de la red esta conectado a cada nodo de la capa siguiente. Esta capa desempeña la función de clasificación basado en los patrones extraídos a través de las capas previas y sus diferentes filtros, es decir, los mapas de características. Estos se convierten en un vector que alimenta dicha red, devolviendo como resultado las clases a las que pertenece la imagen.

En la figura 2.8 se visualiza una estructura genérica de una red convolucional para la tarea de clasificación de dígitos manuscritos entre cero y nueve. La entrada de la red es una imagen en blanco y negro (solo un canal de color) que atraviesa una serie de capas convolucionales y de agrupamiento (en celeste) cuyo resultado alimenta la red densamente conectada (nodos en verde). La salida de la red consiste de nueve nodos (en rojo) correspondiente a un número entre cero y nueve. Aquel que devuelva una probabilidad más alta, será el número identificado en la imagen.

2.5. Fine-Tuning

En el ámbito del aprendizaje profundo, se conoce como *fine-tuning* al proceso de adaptar un modelo previamente entrenado para una tarea o un caso de uso

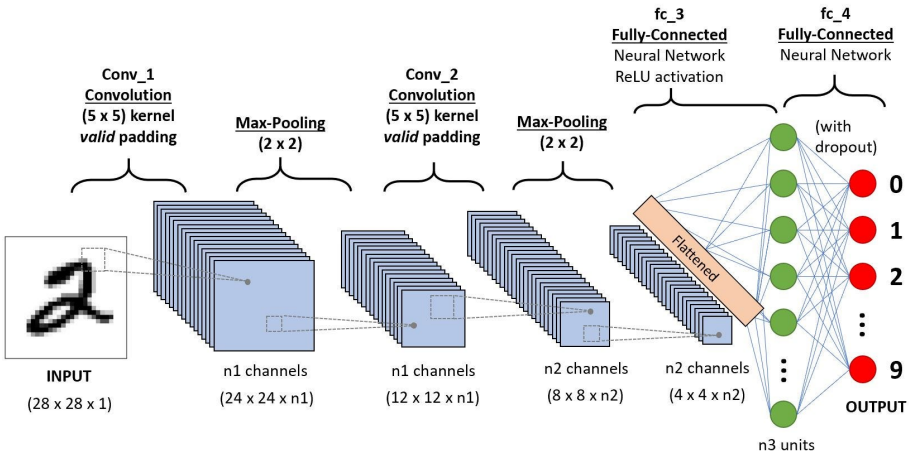


Figura 2.8: Ejemplo de una estructura de una red convolucional. Figura extraída de (Saha, 2018)

específico. Esta técnica es una forma de aprendizaje por transferencia o *transfer learning*, y permite aprovechar el aprendizaje hecho al entrenar un modelo para una tarea en otra tarea especifica relacionada a la primera. Además, permite evitar comenzar el proceso de entrenamiento desde cero, lo cual permite utilizar de manera más eficiente los recursos de computo.

Fine-tuning permite agregar valor al modelo pre-entrenado, reduciendo significativamente el tiempo y recursos requeridos a fin de obtener resultados altamente satisfactorios. Normalmente, la arquitectura del modelo pre-entrenado conserva su integridad durante el proceso. La idea subyacente, es beneficiarse de los patrones y representaciones complejas aprendidas por el modelo en su etapa de entrenamiento, y adaptarlas con el fin de afrontar una tarea más específica. Por añadidura, la obtención de datos etiquetados para un propósito particular entraña un desafío y una inversión de tiempo considerable. La aplicación de *fine-tuning* viabiliza el entrenamiento de modelos sobre conjuntos de datos limitados consiguiendo resultados alentadores con menor esfuerzo (Amanatullah, 2023).

Capítulo 3

Revisión de antecedentes

En este capítulo se hace reseña de investigaciones que se llevaron a cabo en el área de estudio, específicamente siguiendo los enfoques de detección de objetos y de clasificación de imágenes (los cuales se describen en mayor detalle en la sección 2.2). Se seleccionaron distintas investigaciones que aplican diferentes modelos.

Previo a la elaboración de esta sección el equipo realizó una investigación del estado del arte, durante la cual se elaboró un documento del estado del arte. Este se encuentra en (*Proyecto de Grado - Redes neuronales para detección de defectos en saneamiento*, 2024).

3.1. Detección

La detección de objetos es una tecnología del área de visión computacional (*computer vision* en inglés) que consiste en detectar, identificar y en la mayoría de casos delimitar (encuadrar mediante un área rectangular denominada *bounding box* o cuadro delimitador en español) distintos objetos que pueden estar presentes en una imagen o video.

3.1.1. YOLO

Originalmente presentado como resultado de la investigación *You Only Look Once: Unified, Real-Time Object Detection* de 2016 (Redmon, Divvala, Girshick, y Farhadi, 2016), se expone un algoritmo de redes neurnoales convolucionales (CNN, del inglés *convolutional neural network*) y a su vez una familia de modelos que implementan tal algoritmo. Su nombre YOLO, del inglés *You Only Look Once*, alude al hecho de que es capaz de efectuar la tarea de detección con una sola pasada de la red, en contraposición a otros acercamientos que precisan de mayor cómputo para realizar dicha tarea (por ejemplo los métodos de clasificación en dos etapas). A su vez, YOLO se diferencia de sus competidores en que calcula el resultado mediante una única regresión, en contraposición de

otros modelos como Fast R-CNN (otro modelo de redes neuronales para la detección de objetos en imágenes) que utiliza dos salidas, una clasificación para las probabilidades y una regresión para calcular las coordenadas de los cuadros delimitadores (información extraída de (Terven y Cordova-Esparza, 2023)). Hasta la fecha de la creación de este documento, se han desarrollado diez versiones del algoritmo (YOLOv1, YOLOv2, etc). A lo largo de las iteraciones del modelo, se ha intentado ir mejorando tanto la velocidad como la precisión de las predicciones.

La primera versión publicada fue la de YOLOv1. Este modelo, unifica las etapas de la detección de objetos detectando todos los cuadros delimitadores simultáneamente. Para esto, YOLO divide la imagen de entrada en una cuadrícula de S x S y por cada celda de la cuadrícula predice B cuadros delimitadores, y la confianza para C clases diferentes, es decir un conjunto de probabilidades de que un objeto de cada clase esté presente en dicha celda. A su vez, cada predicción de cuadro delimitador consiste de cinco valores: P_c (puntuación de confianza sobre la probabilidad de que contenga un objeto de clase c), b_x y b_y (coordenadas del centro del cuadro, relativas a la celda de la cuadrícula), bh y bw (altura y ancho del cuadro, en relación a la imagen completa). Por tanto, YOLO produce una salida de dimensión S x S x (B x 5 + C). En la figura 3.1 se visualiza un vector de salida simplificado considerando una cuadrícula de tres por tres, tres clases diferentes, y un único cuadro delimitador por celda, resultando en una salida de 3 x 3 x 5.

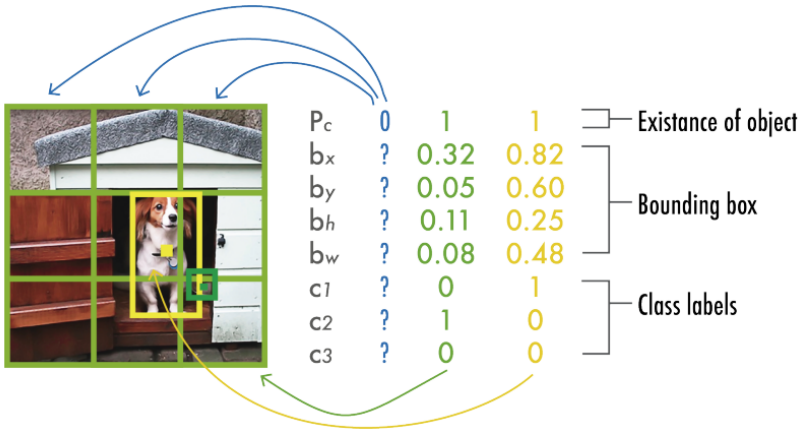


Figura 3.1: Predicción de salida de YOLO para un modelo simplificado. Figura extraída de (Terven y Cordova-Esparza, 2023).

La arquitectura consiste de 24 capas convolucionales seguidas de dos capas densamente conectadas que predicen las coordenadas de los cuadros delimitadores y probabilidades. Todas las capas utilizaron activaciones de unidades lineales rectificadas con fugas, a excepción de la última que empleó una función de activación lineal. Asimismo, YOLO utiliza capas convolucionales 1 x 1 con

el fin de reducir el número de mapas de características y mantener el numero de parámetros bajo.

En este primera versión del algoritmo, se pre entrenan las 20 primeras capas de YOLO con una resolución de imagen de 224 x 224 utilizando el conjunto de datos ImageNet (Russakovsky y cols., 2015). Luego se agregaron las últimas cuatro capas con pesos inicializados aleatoriamente y se afinó el modelo con los conjuntos de datos PASCAL VOC 2007 y VOC 2012 (Everingham, Van Gool, Williams, Winn, y Zisserman, 2010) con una resolución de 448 x 448 para incrementar los detalles para un detección de objetos más precisa. YOLOv1 obtuvo una precisión promedio de 63.4 % en el conjunto de datos PASCAL VOC 2007.

3.1.2. DETR

En la investigación *End-to-end object detection with transformers* (Carion y cols., 2020) se presenta a DETR (Detection Transformer). Este modelo esta diseñado para realizar detección de objetos y está basado en la arquitectura de transformador (Vaswani y cols., 2017).

La arquitectura de transformador se caracteriza por su uso de la autoatención, que valora diferencialmente la importancia de cada parte de la entrada. Esto se logra con una estructura de tipo codificador-decodificador, compuesta por varios submódulos, en particular los módulos de múltiples cabezales de atención son el componente que permiten al modelo aprender varias relaciones y variaciones para elementos de la entrada. En la figura 3.2 se ilustra el funcionamiento general del modelo.

El modelo DETR hace uso de esta arquitectura y a partir de un conjunto de objetos aprendidos logra comprender las relaciones de los objetos y el contexto global de la imagen para realizar predicciones.

En particular, este modelo utiliza una red CNN convencional para aprender una representación 2D de una imagen, a la cual se le agrega información posicional antes de pasárselo al codificador del transformador. Luego el decodificador del transformador procesa una serie de entradas con información posicional y la salida del codificador, y la salida de esta red es finalmente procesada por una red de propagación hacia adelante, que se encarga de la detección de objetos (decir si hay o no un objeto, y en caso de que lo haya dar un cuadro delimitador del mismo).

La arquitectura de DETR tiene la virtud de que reduce la cantidad de componentes que deben ser configurados manualmente. Sin embargo tiene las siguientes debilidades: muestra resultados relativamente malos localizando objetos pequeños. Además, demanda un esquema de entrenamiento considerablemente mayor a otros detectores de objetos modernos. Esto se debe a que la computación de la atención basada en características de la imagen es más complicada de entrenar.

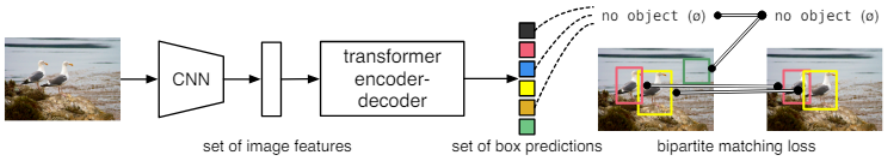


Figura 3.2: Funcionamiento del modelo DETR. Se predicen en paralelo las detecciones combinando una CNN común con una arquitectura de transformadores. Imagen y pie de imagen extraídos de la investigación (Carion y cols., 2020).

3.1.3. Trabajos relacionados

Automatic Detection Method of Sewer Pipe Defects Using Deep Learning Techniques

En este trabajo, se propone un método mejorado basado en el algoritmo YOLOv4 de forma de abordar la problemática de automatizar la detección de defectos. En este sentido, se emplea el módulo de agrupamiento piramidal espacial (SPP, por sus siglas en inglés *spatial pyramid pooling*) el cual expande el campo receptivo y aumenta la habilidad del modelo para fusionar características de contexto en diferentes campos receptivos. Asimismo, se comparan tres funciones de pérdida (o de costo) de cuadros delimitadores GIoU, DIoU y CIoU (del inglés *Generalised Intersection over Union*, *Distance IoU* y *Complete IoU*, respectivamente) en el rendimiento del modelo basado en sus velocidades de procesamiento y precisión en la detección. Adicionalmente, se creó un conjunto de datos que contiene 2700 imágenes y cuatro tipos de defectos.

El flujo de trabajo consistió de cuatro etapas, las cuales fueron la recolección de imágenes de defectos en tuberías de saneamiento y anotaciones de su localización en al imagen; aumento de colección de imágenes; entrenamiento del modelo propuesto; y la detección de defectos y evaluación del rendimiento del sistema.

Para la experimentación se utilizaron imágenes de Sewer-ML, pero dado que este conjunto de imágenes solo tiene las imágenes clasificadas y no presentan cuadros delimitadores, para poder entrenar al modelo fue necesario realizar estos cuadros a mano. Para esto el equipo de la investigación fue ayudado por profesionales del área, obteniendo así un total de 2700 imágenes con cuadros delimitadores.

Entrando a la fase misma de experimentación, se llevaron a cabo tres experimentos para tres propósitos distintos: utilizando YOLOv4, estudiar el impacto de las tres funciones de pérdidas de cuadros limitadores en el desempeño de la detección de defectos, para así seleccionar la función de pérdida que reporte mayor efectividad; basado en el experimento anterior, introducir el módulo SPP para mejorar la estructura de la red. Luego, comparando el desempeño en la detección de las distintas funciones de pérdida en combinación con el módulo SPP, seleccionar el mejor modelo de detección; comparar el modelo resultado del

experimento anterior con otros modelos estado del arte y evaluar la efectividad del modelo desarrollado.

En el primer experimento se utilizaron GIoU, DIoU y CIoU para el cálculo de pérdida de regresión del cuadro delimitador de predicción. Los resultados experimentales muestran que el modelo con mayor precisión promedio en la detección de defectos es la combinación de YOLOv4 con la función de pérdida DIoU (YOLOv4-DIoU) seguido de YOLOv4-CIoU.

Basándose en los resultados del experimento 1, se toman en el segundo experimento a los dos mejores modelos del experimento anterior, YOLOv4-DIoU y YOLOv4-CIoU, puesto que difieren por muy poco, y se procede a mejorar ambos modelos introduciendo el módulo SPP, YOLOv4-D-SPP3 y YOLOv4-C-SPP3, respectivamente.

Por último, se realiza una comparativa en cuanto a los valores de las métricas de exhaustividad y F1 de todos los modelos que se visualizan en la tabla 3.1. En particular, se busca priorizar la medida de exhaustividad debido a que se intenta identificar tantos defectos como sea posible.

Methods	Recall (%)				Average (%)		
	Crack	Deposition	Root	Stagger	Recall	Precision	F1
SSD	73.5	84.5	82.9	91.1	83.0	76.2	79.5
Faster R-CNN	83.8	86.9	85.4	96.2	88.1	49.0	62.8
YOLOv3	70.6	85.7	74.4	91.1	80.5	82.5	81.5
Improved YOLOv3	69.1	83.3	89.0	93.7	83.8	80.8	81.8
YOLOv7	77.9	78.6	81.7	87.3	81.4	90.9	85.5
YOLOv8	72.1	82.1	85.3	98.7	84.5	79.5	81.7
DETR	79.4	90.4	91.4	94.9	89.0	57.9	70.0
Swin-Trans-YOLOv4	33.8	79.7	70.7	87.3	67.8	83.5	73.0
YOLOv4-D-SPP3	83.8	88.1	90.2	93.7	89.0	90.1	89.5

Tabla 3.1: Comparación de métricas de distintos modelos

En conclusión, se presentó un modelo mejorado del algoritmo YOLOv4, YOLOv4-D-SPP3, el cual puede detectar de forma efectiva los distintos tipos de defectos y determinar precisamente su localización en la imagen, obteniendo una precisión promedio del 92.3 % y una exhaustividad promedio del 89.0 %, situándolo como uno de los mejores modelos en comparación con otros que son estado del arte.

DefectTR: End-to-end defect detection for sewage networks using a transformer

En este trabajo se presenta un método para la detección de defectos en imágenes de saneamiento basado en la arquitectura transformador de detección (DETR). A su vez, se presenta un enfoque para el análisis de la severidad de los defectos basado en la capacidad de autoatención de los transformadores.

El sistema propuesto DefectTR tiene tres principales procesos: preprocesamiento, detección basada en transformador y análisis de severidad de defectos.

En la fase de preprocesamiento las imágenes del conjunto de datos son procesadas para reducir problemas como el brillo desparejo o la bruma.

La primera técnica que se aplica es la conocida como DHE (Abdullah-Al-Wadud, Kabir, Akber Dewan, y Chae, 2007). Con esto se logra mejorar el contraste de las imágenes sin perder detalles. Además, existe el problema de que las imágenes pueden verse nubladas por la humedad y el vapor que hay en el saneamiento. Para reducir esto se aplica un modelo de reducción de ruido preentrenado llamado GCANet (Chen y cols., 2019).

En la fase de detección basada en transformador se utiliza una versión modificada del modelo DETR (Carion y cols., 2020) denominado DefectTR.

Para abordar las limitaciones de DETR, en DefectTR se plantean los siguientes cambios:

- ColorJitter: Color jitter es un método de aumento que cambia de manera aleatoria el brillo, la tonalidad y la saturación de una imagen. El rango de valores para estos parámetros no puede ser muy grande para no introducir ruido en los datos.

- Función de activación LeakyReLU: La función unidad lineal rectificada (ReLU) usualmente sufre de un problema donde las neuronas se trancan en el lado negativo y siempre retornan 0. Estas neuronas se vuelven inutilizables ya que no cumplen ningún rol aprendiendo características. Se propone el uso de LeakyReLU, que elimina las partes de pendiente cero de ReLU, lo cual también mejora el tiempo de entrenamiento.

- Optimizador LaProp: El optimizador Adam es el algoritmo de descenso por gradiente estocástico que se utiliza en DETR. Se cambia este algoritmo por el optimizador LaProp que permite separar los parámetros de momento y adaptividad. Los resultados mostraron que LaProp obtiene mayor velocidad y estabilidad que Adam en varios conjuntos de datos.

- Función de costo *Complete Intersection over Union* (CIoU): La función de costo *Generalized Intersection over Union* (GIoU) que es utilizada por el modelo DETR, expande el cuadro delimitador predicho para incluir al cuadro delimitador de verdad absoluta. Se propone utilizar CIoU en vez de GIoU porque, en contraste con GIoU, se ajusta el cuadro predicho precisamente sobre el cuadro de verdad absoluta. Se probo que CIoU converge más rápido que la función de costo GIoU.

  El sistema fue construido y entrenado utilizando PyTorch, una biblioteca de aprendizaje automático de código abierto.

Para la elaboración de este trabajo se creó un conjunto de datos con imágenes de distintas tuberías de saneamiento hechas de concreto tomadas en Corea del Sur. Las imágenes fueron capturadas por robots equipados con cameras SONY Exmor CMOS que pueden rotar 360°.

Las imágenes fueron etiquetadas según los siguientes 10 tipos de defectos por un grupo de profesionales de la inspección de saneamiento:

- Rotura (BK)
- Grieta longitudinal (LC)
- Grieta circunferencial (CC)
- Escombros (DS)
- Unión desplazada (DJ)
- Unión defectuosa (FJ)
- Unión desplazada (FJ)
- Lateral saliente (PL)
- Daño en la superficie (SD)
- Intrusión de raíz (RI)

Se realizaron varios experimentos utilizando el conjunto de datos generado para evaluar el rendimiento de DefectTR en distintos escenarios.

Para evaluar el rendimiento del pre-procesamiento de imágenes propuesto se analizaron los resultados del método al aplicar el procesado, en comparativa con los resultados obtenidos sin aplicarlos. Las pruebas fueron realizadas usando el conjunto de datos generado. Los resultados, los cuales se muestran en la tabla 3.2 reafirman que este paso tiene un rol crucial en la mejora del rendimiento en la detección de defectos en saneamiento.

Dataset	Approach	Precision (%)	Recall (%)	mAP (%)
Original	SSD [43]	49.4	56.3	41.7
	YOLOv4 [18]	57.3	60.6	46.8
	Faster R-CNN [19]	58.8	62.7	53.4
	CenterNet [44]	59.3	64.3	48.7
	DETR [17]	56.5	66.1	54.5
	DefectTR	58.7	68.2	55.9
Pre-processing	SSD [43]	53.1	55.3	43.9
	YOLOv4 [18]	57.9	65.7	47
	Faster R-CNN [19]	57.5	63.2	56.2
	CenterNet [44]	59.7	60.1	51.9
	DETR [17]	61.2	69.3	54.2
	DefectTR	65.4	69.7	**60,2**

Tabla 3.2: Resultados experimentales del pre-procesamiento

Luego se realizaron experimentos para evaluar las mejoras propuestas a DETR. Se crearon 4 modelos, donde se agregan gradualmente las modificaciones mencionadas. En la tabla 3.3 se muestran describen los modelos junto a la métrica mAP (Precisión media en promedio) que evalúa el desempeño en promedio para todas las clases.

	ColorJitter	Laprop	LeakyRelu	CIoU	mAP
Model A	✓				56.2
Model B	✓	✓			56.2
Model C	✓	✓	✓		57.8
Model D	✓	✓	✓	✓	60.2

Tabla 3.3: Modelos utilizados para probar las modificaciones propuestas

Se observo que en el modelo A se obtuvo un desempeño un 5 % mejor que original. Tras evaluar todos los modelos se concluyo que con estas mejoras se aumento el mAP obtenido respecto al modelo DETR.

Model	Precision (%)	Recall (%)	mAP	Inference time (ms)
SSD	53.1	55.3	43.9	**36**
YOLOv4	57.9	65.7	47	48
Faster-RCNN	57.5	63.2	56.2	1000
CenterNet	59.7	60.1	51.9	128.2
DETR	61.2	69.3	54.2	83
DefectTR	**65**,**4**	69.7	60.2	85

Tabla 3.4: Resultados de experimentos sobre métodos estado del arte

Finalmente se realizó un experimento para comparar el rendimiento de DefectTR con otros modelos estado del arte, los cuales incluyen SSD, YOLOv4, Faster R-CNN, CenterNET y DETR, utilizando el conjunto de datos recolectado. Dichos resultados se observan en la tabla 3.4, donde se observa como DefectTR obtuvo el mejor mAP con un 60.2 %. El tiempo de inferencia fue de 85ms por imagen, mientras que SSD logra un tiempo de 36ms.

En conclusión, se introduce un nuevo método basado en la arquitectura de transformadores para la detección de defectos en saneamiento mediante imágenes. Se introducen varios cambios, incluyendo la función de activación Leaky-ReLU, el optimizador LaProp y la función de costo CIoU para mejorar el rendimiento del modelo DETR. Además, se observo que el uso de los pesos de atención del modelo pueden ser utilizados para el análisis de la severidad de los defectos.

3.1.4. Conjuntos de datos disponibles

El número de conjuntos de datos disponibles para detección de objetos en el dominio de este trabajo es limitado. Los conjuntos de datos más completos

se encontraron en el sitio Roboflow (Roboflow, s.f.). Puntualmente, el conjunto (Detection, 2023) y el conjunto (Sewage, 2024).

Estos conjuntos de datos no superan las 1.000 imágenes, además de no asegurar la calidad de las clasificaciones ya que no se especifica que los conjuntos hayan sido creados por profesionales del área. Además, la clasificación utilizada en ambos no sigue la especificada por NASSCO.

3.2. Clasificación

La clasificación de imágenes es otra tecnología del área de visión computacional, que consiste en clasificar imágenes dado un conjunto finito de clases, del cuál se puede clasificar tanto con una (*clasificación binaria*) o más (*clasificación multi-etiqueta*) categorías (2.2.1). La idea es poder clasificar una imagen tanto en base a lo que esta representa (*binaria*), como también a distintos elementos que pueda contener (*multi-etiqueta*).

3.2.1. ResNet

En los intentos de mejora en el ámbito de la visión computacional se han intentado diferentes abordajes, pero lo que se ha establecido como estándar es el uso de redes neuronales para hacerlo. Estas redes exigen un alto poder de cómputo a medida que se incrementa el número de sus capas, lo cual posibilita un reconocimiento eficaz de características o patrones relevantes. No obstante, una red profunda entraña la problemática del desvanecimiento del gradiente. Este problema se produce cuando se realiza la propagación hacia atrás para calcular el gradiente de la función de pérdida con respecto a los pesos de la red, para luego ajustar dichos pesos y minimizar la pérdida. Sin embargo, a medida que se propaga el gradiente hacia atrás en una red (desde la capa de salida hacia las capas más cercanas de la entrada) puede ocurrir que su valor vaya disminuyendo, hecho que se ve agravado si la red cuenta con un gran número de capas, impidiendo así que se ajusten correctamente los valores de los pesos. Para solucionar este problema, en la investigación *Deep Residual Learning for Image Recognition* (He, Zhang, Ren, y Sun, 2016) se presenta una arquitectura de redes residuales.

Las redes neuronales residuales (ResNet, del inglés *Residual network*) son modelos de aprendizaje profundo utilizados para tareas de visión computacional, cuya particularidad es el uso de conexiones "residuales". Estas conexiones permiten que las entradas se sumen directamente a las salidas de ciertas capas, lo que a su vez permite que la información original se "salte" algunas capas en vez de pasar a través de ellas completamente (como se muestra en la figura 3.3). Esto resultó ser muy efectivo en la tarea de mitigar el problema del desvanecimiento del gradiente, lo que posibilitó el uso de redes neuronales más profundas sin que ese problema sea un impedimento determinante.

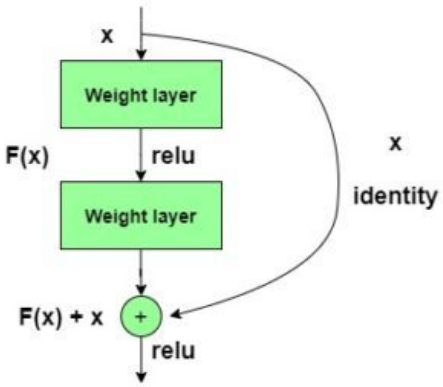


Figura 3.3: Representación del funcionamiento de una red neuronal residual. Figura extraída de (jaideepsinghsandhu - GeeksForGeeks, 2020)

3.2.2. DenseNet

En la investigación Densely Connected Convolutional Networks (Huang, Liu, van der Maaten, y Weinberger, 2018) se presenta a DenseNet, un modelo de redes neuronales que se destaca por contar con bloques densos (de ahí el nombre). Estos bloques se componen de varias capas densas o *fully connected*, donde todos los nodos de cada capa se conectan con todos los nodos de la siguiente. Adicionalmente, cuenta con la particularidad de que existen conexiones (hacia adelante, sin crear ciclos) entre capas no adyacentes.

3.2.3. ViT

El desarrollo del modelo ViT se fundamenta en principios derivados de otra rama del aprendizaje profundo, el del procesamiento del lenguaje natural. En la investigación donde se presenta al modelo (Dosovitskiy y cols., 2020), se da un poco de contexto que puede ser de ayuda a la hora de entender el paso del PLN a la clasificación de imágenes: se utilizan transformers, modelos que cuentan con *capas de atención*, las cuáles se encargan de enfocar la atención (valga la redundancia) del modelo hacia ciertos puntos específicos del objeto a ser analizado, como lo puede ser por ejemplo una oración en caso del procesamiento del lenguaje natural.

Llevándolo al caso de estudio, la atención se puede transcribir del PLN a la clasificación de imágenes cambiando el ejemplo de una oración por el de una imagen. Se aplican transformadores a las imágenes de forma directa, dividiendo en el proceso a estas imágenes en fragmentos.

Este modelo ha sacudido por completo al estado del arte, tanto es así que buena cantidad de los modelos estado del arte actuales son derivados de este modelo y utilizan transformers (PapersWithCode, s.f.).

3.2.4. DaViT

Este modelo (Ding y cols., 2022) se basa en el de Vision Transformer (ViT - 3.2.3), mas tiene la particularidad de que se implementan dos focos de atención en vez de uno como en el original.

3.2.5. Inception

El modelo Inception (Szegedy y cols., 2015) es otro modelo de redes neuronales, cuyo éxito se formó a partir de su construcción basada en la optimización de recursos computacionales. En la investigación se comenta que para la creación del modelo se tuvo que pasar por una fase de prueba y error donde meticulosamente se fue ideando y cambiando el diseño para no perder la optimización de los recursos (se muestra un ejemplo en la figura 3.4). Gracias a esto, se pudo crear una red de mayor profundidad sin aumentar el tiempo de cómputo.

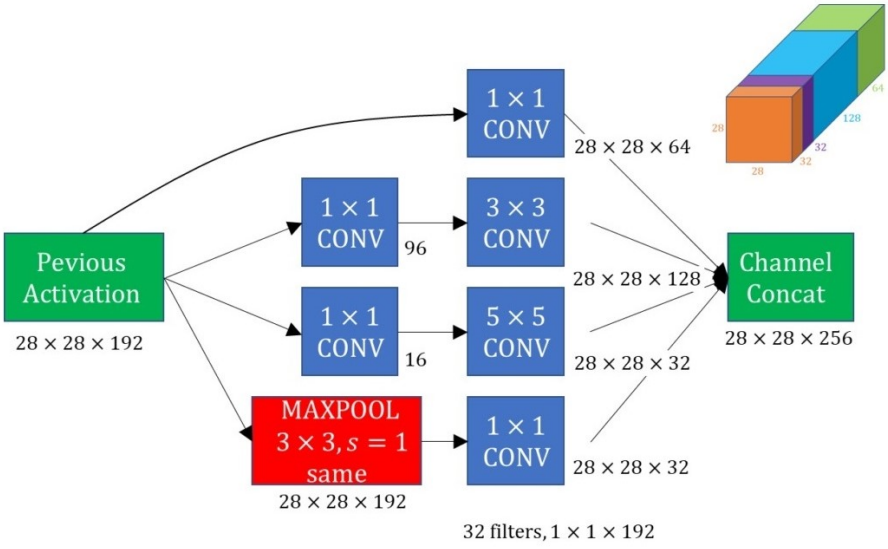


Figura 3.4: Ejemplo de un módulo Inception. Figura extraída de (*CNN Inception Network*, s.f.).

3.2.6. Trabajos relacionados

Automatic Detection and Classification of Sewer Defects via Hierarchical Deep Learning

Existen varios obstáculos a la hora de afrontar la problemática de la detección de defectos en tuberías de saneamiento, tales como la dificultad para conseguir

imágenes (debido a la inconveniencia intrínseca de manipular tuberías que se encuentran varios metros debajo del suelo), o la dificultad a la hora de categorizar distintos tipos de defectos, que no necesariamente son solamente de un tipo u otro, además de haber casos de defectos sumamente parecidos. Sin embargo, uno de los inconvenientes más importantes surge a la hora de llevar esta tarea hacia la automatización, dado que en lo general las tuberías se encuentran sin defectos notables, cosa que se ve fuertemente reflejada en los conjuntos de datos conseguidos a la hora de la recolección de datos para el entrenamiento de redes neuronales. Según esta investigación hecha por (Xie, Li, Xu, Yu, y Wang, 2019), aproximadamente la mitad de las imágenes conseguidas entran dentro de la categoría "sin defectos", mientras que el resto de las categorías se repartían entre la otra mitad, también de forma desbalanceada 3.5. El tener un conjunto de datos desbalanceado es un problema conocido en el ámbito de las redes neuronales, ya que afecta negativamente los resultados de estas al no tener una cantidad proporcional de datos para entrenar a cada categoría dispuesta. En este artículo se intenta entonces resolver este problema.

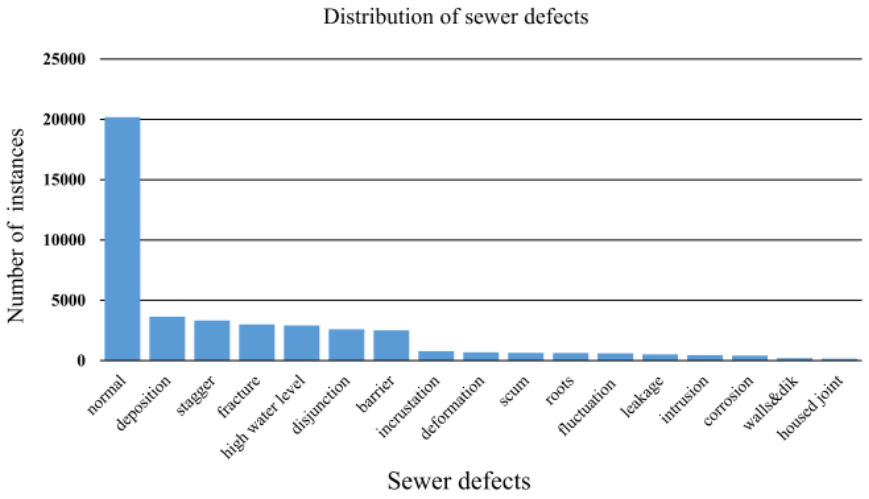


Figura 3.5: Clases de defectos encontrados. Figura extraída del artículo (Xie y cols., 2019).

Entrando en la descripción del método, en esta investigación se utiliza la arquitectura CNN, lo cual no sorprende teniendo en cuenta lo bien que se adapta a problemas de evaluación de imágenes. La particularidad de esta investigación es la forma en la que se utiliza la arquitectura de esta red neuronal, ya que se optó por crear una arquitectura jerárquica donde haya dos CNNs, y se utilicen una después de la otra. Como se mencionó previamente, un gran problema a resolver es el de los conjuntos de datos desbalanceados. Se mencionó también

que aproximadamente la mitad de las imágenes recolectadas eran clasificadas como “sin defectos”, lo que nos deja otra mitad aproximadamente de imágenes de tuberías “con defectos”, entonces los investigadores llegaron a la conclusión de que lo primero que se podría hacer es construir una red neuronal donde solamente se detecten defectos. Esto tiene sentido, ya que un clasificador binario que se entrene a partir de un conjunto de datos evaluados aproximadamente 50/50 en ambas clasificaciones suena muy prometedor, y así lo fue.

A priori resuelto el problema de la detección, queda el problema de la clasificación, y aquí es donde entra la segunda CNN. Si bien existe un desequilibrio en la categorización dentro del subconjunto de imágenes clasificadas como “con defectos”, se pueden usar redes neuronales para su clasificación (de tipo de defectos, no de la presencia de los mismos). Se utiliza entonces otra CNN para su clasificación, lo cuál puede traer consigo un inconveniente: para esta red neuronal solo se podría utilizar la mitad (aproximadamente) del conjunto de datos extraído, lo cuál sería desaprovechar la mitad de los datos, además de aumentar el peligro de entrenar a la red con un escaso número de ejemplos dentro del conjunto de datos. Para solucionar esto, se utilizan los pesos calculados de la red neuronal anterior (básicamente reutilizar la CNN anterior), aprovechando así el entrenamiento previo (aprendizaje por transferencia, o *transfer learning*).

En cuanto a los datos utilizados, para esta investigación se tomó un conjunto de datos de 4000 imágenes de sistemas de cañerías, las cuáles fueron manualmente etiquetadas por profesionales del área. Como información adicional a detallar, este conjunto de datos se hizo público en conjunto a la investigación.

Los resultados en general son bastante positivos, mostrándose así en la tabla de resultados obtenidos:

Metrics	**2 Classes**	**6 Classes**						
	Defect	FR	HW	DE	DI	ST	BA	Avg.
Accuracy	0.945	0.9578	0.9456	0.9433	0.9561	0.9428	0.9522	0.9496
Presicion	0.9684	0.8784	0.8389	0.8173	0.885	0.8339	0.8543	0.8513
Recall	0.92	0.8666	0.8333	0.85	0.8467	0.80	0.86	0.8461
F1-score	0.9436	0.8725	0.8361	0.8334	0.8654	0.8269	0.8571	0.8486

Tabla 3.5: Evaluación cuantitativa de los modelos, incluyendo dos niveles de discriminación jerárquica. Tabla extraída del artículo (Xie y cols., 2019).

Como se puede ver en la tabla 3.5, se consigue una exactitud promedio de 94.96 %. Si bien las demás métricas no consiguen un nivel de acierto tan marcado como el de la exactitud, tampoco se consideran valores bajos.

En conclusión, esta investigación arroja luz sobre una solución posible a una problemática ya conocida acerca de los conjuntos de datos del área, la cuál es el desequilibrio de los mismos (siempre va a haber una gran desproporción de imágenes clasificadas como no defectuosas por sobre las defectuosas). El concepto de jerarquizar las CNNs puede ser de ayuda potencialmente a priori para cualquier investigación del rubro. La idea a su vez se retroalimenta a sí

misma para la CNN de segunda jerarquía.

3.2.7. Conjuntos de datos disponibles

Hasta la fecha en la que se escribe este documento, no se dispone de un amplio número conjunto de datos. Sin embargo, el equipo se encontró con un conjunto de imágenes de libre uso (para uso académico) de un tamaño suficiente.

En el trabajo expuesto en (Haurum y Moeslund, 2021) se presenta un conjunto de datos *multi-etiqueta* denominado Sewer-ML, que consiste de 1.3 millones de imágenes anotadas o etiquetadas por inspectores profesionales en sistemas de saneamiento. Se pretende adentrar en las particularidades que componen a este conjunto, exhibiendo su proceso de construcción.

Los datos fueron obtenidos a partir de más de 75.000 videos de inspecciones de tuberías de saneamiento recolectados por distintas compañías de servicios públicos de Dinamarca, en el período 2011-2019. Estos videos fueron etiquetados por inspectores sanitarios conforme a un estándar danés que comprende 18 clases específicas. Cada clase, a su vez, tiene asociado una puntuación (normalizada en el intervalo [0,1]) que representa el peso de cada clase (CIW, por sus siglas en inglés *class-importance weight*). Este peso se basa en el costo económico que conlleva la reparación de dicho defecto. En la tabla 3.6 se pueden observar las distintas clases junto con su puntuación. Asimismo, estos datos contemplan una gran variedad de materiales, formas y dimensiones de tuberías.

Code	Description	CIW
VA	Water Level (in percentages)	0.0310
RB	Cracks, breaks, and collapses	1.0000
OB	Surface damage	0.5518
PF	Production error	0.2896
DE	Deformation	0.1622
FS	Displaced joint	0.6419
IS	Intruding sealing material	0.1847
RO	Roots	0.3559
IN	Infiltration	0.3131
AF	Settled deposits	0.0811
BE	Attached deposits	0.2275
FO	Obstacle	0.2477
GR	Branch pipe	0.0901
PH	Chiseled connection	0.4167
PB	Drilled connection	0.4167
OS	Lateral reinstatement cuts	0.9009
OP	Connection with transition profile	0.3829
OK	Connection with construction changes	0.4396

Tabla 3.6: Clases de inspección de tuberías y el peso asociado a cada una (CIW, de *class-importance weight*). Tabla extraída del artículo de Sewer-ML. (Haurum y Moeslund, 2021)

El proceso de construcción del conjunto consiste en extraer un único cuadro (*frame*) para cada anotación de defecto en un video de inspección de tubería. Es decir, cada vez que exista una anotación de una clase particular presente en el video, se extrae un único *frame* correspondiente a dicha anotación y se lo incluye en el conjunto de datos. Cada anotación corresponde a una anotación de verdad absoluta (*ground truth annotation*) de una clase singular en un segundo específico del video, junto con una locación asociada dentro de la propia tubería. La representación *multi-etiqueta* se obtiene combinando aquellas anotaciones que están cerca una de la otra en la tubería.

De las 18 clases mencionadas, no todas comprenden instancias de defectos en tuberías, sino que también pueden indicar información relevante, como por ejemplo, un cambio en la forma o material de la tubería, la presencia de una ramificación o conexiones de tuberías. La clase VA que se visualiza en la tabla 3.6 es una clase especial dado que esta anotada al comienzo y final de un video de inspección así como también cuando el nivel del agua cambia dentro de un intervalo de paso del 10 % (cuando se supera el 10 %). Esto implica que todas las anotaciones poseen un nivel de agua asociado. Adicionalmente, mediante la aplicación de una serie de heurísticas, se obtienen observaciones de casos sin clases anotadas, denominados no defectuosos (ND, por sus siglas en inglés). Las anotaciones de clase VA son agrupadas junto con las de ND en caso de que no existir otras etiquetas que ocurran simultáneamente. Esto resulta en un total de 690.722 imágenes de tuberías de saneamiento "normales" (no defectuosas) sin clases anotadas, y 609.479 imágenes con una o más clases anotadas denominadas "defectuosas".

Por tanto, el problema de clasificación *multi-etiqueta* consiste en la predicción de las etiquetas de las clases de la tabla 3.6, a excepción de la clase VA. Esto implica que una tubería sin anotaciones de clases representa la ausencia de cualquiera de ellas; esto es por consecuente, una clase implícita.

Respecto a la división del conjunto, se separan los datos en tres particiones: entrenamiento (*training*), validación (*validation*) y prueba (*test*). Luego, se seleccionan aleatoriamente videos hasta que el 80 % de todas las anotaciones estén en la partición de entrenamiento, y el restante 20 % es separado en partes iguales entre validación y prueba. Esto resulta en más de 60.000 videos en la partición de entrenamiento, y más de 7.000 en cada una de las particiones de validación y prueba. De esta forma, se asevera que ninguna imagen de la misma tubería esté presente entre distintas particiones. Estas particiones conducen a una división casi uniforme de observaciones normales y defectuosas como se aprecia en la tabla 3.7.

Observando la distribución de ocurrencias de clases en la figura 3.6, se constata que están uniformemente representadas en cada partición, lo cual sugiere una distribución de clase similar entre cada una. No obstante, es notorio el sesgo existente hacia algunas de las clases como "Normal" y "FS". Este suceso, expone el desbalance presente en el conjunto de datos, representativo de la distribución de clases en el mundo real.

Type	Training	Validation	Test	Total
Normal	552,820	68,681	69,221	690,722
Defective	487,309	61,365	60,805	609,479
Total	1,040,129	130,046	130,026	1,300,201

Tabla 3.7: Número de imágenes que contienen observaciones normales y defectuosas en las tres particiones. Tabla extraída del artículo de Sewer-ML (Haurum y Moeslund, 2021).

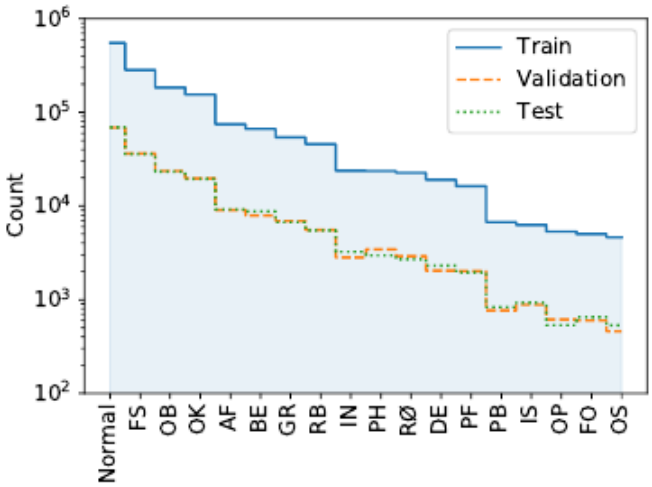


Figura 3.6: Frecuencias de las clases anotadas y la clase normal. Figura extraída del artículo de Sewer-ML (Haurum y Moeslund, 2021).

Capítulo 4

Desarrollo del proyecto

En este capítulo se documenta el proceso de elaboración del proyecto. Primeramente, se describen las etapas de investigación donde se dictamina el enfoque a seguir.

A continuación, se dedica una sección a la tarea de clasificación de imágenes, ahondando en los conjuntos de datos y modelos seleccionados, y detalles de su implementación. Luego de esta, una sección de experimentación en la que se elucida sobre la infraestructura empleada para la ejecución de pruebas, métricas seleccionadas para la evaluación de modelos, así como también la metodología seguida de pruebas y una reseña de los mejores modelos obtenidos.

Por último, se presenta el desarrollo de un sistema para DICA & Asociados, profundizando en sus etapas de diseño, arquitectura e implementación.

4.1. Etapas del proyecto

El proyecto se descompone en dos etapas. Durante la primera, se trabajó en encontrar los mejores modelos para la detección de defectos en saneamiento. En este sentido, se llevó a cabo una extensa investigación sobre el estado del arte, donde surgieron dos enfoques posibles a fin de abordar el problema en cuestión: la clasificación de imágenes *multi-etiqueta* y la detección de objetos. La diferencia entre estos dos enfoques se describe más en detalle en la sección 2.2.

Se tomó la decisión de trabajar con modelos pre-entrenados con conjuntos de datos de propósito general, aplicando la técnica de *fine-tuning* (descrita en la sección 2.5) sobre conjuntos de datos de defectos en saneamiento. Este enfoque fue necesario para obtener modelos de detección de defectos en saneamiento, dado que los trabajos consultados no disponibilizan los pesos de los modelos que generan.

En cuanto al enfoque a seguir, se optó por centrar la investigación en modelos de clasificación de imágenes, resolución que aflora tras realizar pruebas iniciales con modelos de detección de objetos YOLO y DETR, y detectar que los con-

juntos de datos de defectos disponibles para estos, contenían pocas imágenes (entorno a las mil cada uno), con la condicionante de que el etiquetado no era fiable en todos los casos. Es pertinente destacar que dicho etiquetado consiste no solo del tipo o clase de defecto, sino que también del cuadro delimitador (*bounding box*) que lo identifica en la imagen. La generación de un nuevo conjunto de imágenes de este estilo, constituye una tarea demandante de tiempo y extenso conocimiento acerca del dominio, promoviendo su descarte del alcance del proyecto. Por otra parte, para la clasificación de imágenes se dispone del conjunto de datos Sewer-ML (Haurum y Moeslund, 2021) que contiene más de 1.170.000 imágenes etiquetadas por profesionales. Asimismo, se opta por un problema de carácter multi-etiqueta debido a la posible presencia de varios defectos en una imagen.

El equipo definió una metodología experimental a seguir que consistió en elegir ciertos modelos a probar y realizar pruebas iniciales con ellos. Estos modelos fueron seleccionados a partir de la investigación sobre el estado del arte previamente aludida. Luego, en base a los resultados obtenidos en estas pruebas, se extendió incrementalmente la experimentación con los modelos que obtuvieron mejores resultados y margen de mejora.

En la sección 4.2 se describe sobre la aplicación de modelos de clasificación de imágenes a la problemática de detección de defectos en saneamiento. Se profundiza sobre los conjuntos de datos utilizados, los modelos que fueron seleccionados para experimentar y la implementación de programas para su entrenamiento. Posteriormente, se describen los experimentos realizados, la infraestructura utilizada, las métricas consideradas, la metodología que se siguió y los resultados obtenidos. Esto se encuentra en la sección 4.3.

En esta primera etapa, el equipo también participó de una jornada de trabajo de campo del grupo MINA. Este evento, permitió al equipo tener un acercamiento al ambiente en el que operan los robots de inspección junto con profesionales de la empresa DICA & Asociados. También se conoció más sobre la línea de investigación del grupo MINA y otros proyectos en desarrollo.

En la segunda etapa del proyecto, el equipo se dedicó al desarrollo de un sistema que permite a la empresa DICA & Asociados hacer uso de los modelos generados. El sistema emplea los modelos entrenados para el procesamiento de videos de inspección de tuberías recolectados por la empresa, y ofrece dos funcionalidades: la generación de un documento que reporte los defectos detectados en el video-inspección, clasificándolos según su tipo, y un nuevo video (video enriquecido) que exhibe la variación de las predicciones del modelo, pudiendo identificar visualmente, los instantes del video-inspección que presentan mayor concentración de defectos. Ambas funcionalidades son accesibles desde una interfaz gráfica, la cual permite al usuario seleccionar el modelo que desee utilizar, así como cambiar los parámetros de la generación de reportes y el video enriquecido.

Figura 4.1: Trabajo de campo en el colector del puerto del buceo

Figura 4.2: Trabajo de campo en el colector del puerto del buceo

4.2. Clasificación de imágenes

Se decidió seguir el enfoque de la clasificación de imágenes para la detección de defectos presentes en cada imagen. Al margen de esta decisión, se busca obtener modelos que realicen una clasificación *multi-etiqueta*, donde cada defecto posible es una clase. Estos modelos dan como salida un valor que representa la confianza en la predicción de cada defecto en la imagen procesada.

4.2.1. Conjuntos de datos utilizados

Para el entrenamiento de los modelos de clasificación de imágenes se requiere de un conjunto de datos compuesto por imágenes y etiquetas. Particularmente, para el dominio de trabajo, se precisan imágenes tomadas de inspecciones de saneamiento, y para cada una, conocer qué defectos están o no presentes.

En general, el tipo de modelos basados en redes neuronales empleados en este trabajo, se benefician de ser entrenados con un gran volumen de datos. Por este motivo, se busca entrenar a los modelos con un gran conjunto de datos que les permita aprender a reconocer los patrones o representaciones que diferencian a los distintos tipos de defecto.

Se utilizó como base el conjunto de datos Sewer-ML (Haurum y Moeslund, 2021), el cual se describe en mayor detalle en la sección 3.2.7. A partir de dicho conjunto, se crean nuevos subconjuntos de distintos tamaños, seleccionando los defectos de interés, con la intención de experimentar con diversos modelos y evaluar los resultados que estos reportan, conforme se incrementa el volumen de datos. A continuación se detallan las diferencias entre dichos subconjuntos.

Defectos seleccionados

El conjunto de datos Sewer-ML contiene un etiquetado de defectos que no tienen una correspondencia uno a uno con los defectos especificados en el programa PCAP de NASSCO (véase 2.1). Además, no todos los defectos etiquetados son utilizados para las inspecciones realizadas por la empresa DICA & Asociados.

Por estos motivos, se realizó una selección de defectos de Sewer-ML a ser utilizados junto con una correspondencia de estos defectos con la clasificación del programa PCAP. Estos se muestran a continuación.

- RB - Grietas, roturas y colapsos (corresponde a los defectos Grieta, Fractura, Rotura y Colapso del PACP)
- FS - Junta desplazada (corresponde al defecto Junta del PACP)
- AF/BE - Depósitos o sedimentos (corresponden al defecto Depósito del PACP)
- RO - Raíces (corresponde al defecto Raíces del PACP)
- PH - Conexión cincelada (si bien este defecto no tiene una correspondencia directa con un defecto del PCAP, fue considerado de interés por DICA para la detección de intrusiones)

- FO - Obstáculo (corresponde al defecto Obstáculo del PACP)

Conjuntos de distintos tamaños

La experimentación con diferentes modelos es un proceso que conlleva un tiempo considerable, incluso cuando se dispone de hardware específico para la tarea. Por este motivo, se decidió crear conjuntos de distintos tamaños que permiten experimentar entrenando modelos con menos imágenes. De esta manera se logra iterar con mayor velocidad en los experimentos.

Los conjuntos se crearon a partir del conjunto de datos obtenido en la selección de los defectos o etiquetas descritas en la sección previa. Estos subconjuntos son generados a partir de una selección aleatoria de dicho conjunto.

Puntualmente se generaron tres subconjuntos de 120.000, 500.000 y 1.170.000 imágenes cada uno.

Subconjuntos de entrenamiento, validación y prueba

Cada conjunto de los mencionados en la sección anterior está dividido, a su vez, en tres subconjuntos: entrenamiento, validación y prueba. La división en este formato es una practica común cuando se trabaja con modelos de aprendizaje profundo, que permite el entrenamiento de modelos evitando el sobre-ajuste y mantener un conjunto separado para las pruebas. En el anexo A se describe en mayor detalle el uso de estos subconjuntos, y los motivos por los que son necesarios para mantener rigurosidad al experimentar.

4.2.2. Modelos seleccionados

Partiendo de la información relevada en la investigación del estado del arte, se eligieron modelos para ser entrenados con los conjuntos de datos generados. Se revisaron, por un lado, modelos que obtuvieron buen rendimiento en trabajos anteriores con imágenes de defectos en saneamiento, particularmente, se consideraron las reseñas del estado del arte de (Li, Wang, Dang, Song, y Moon, 2022) y (Haurum y Moeslund, 2021). Por otro lado, se consideraron, asimismo, modelos del estado del arte en clasificación de imágenes en general, como los que se presentan en (PapersWithCode, s.f.)

El equipo centró esfuerzos en generar modelos que pudieran ser utilizados por la empresa DICA & Asociados. Para ello, se optó por buscar modelos cuyos requerimientos para realizar inferencias en un marco de tiempo razonable no suponga un obstáculo para el hardware de oficina moderno.

En este sentido, la principal limitante es la memoria RAM del hardware, debido a que la inferencia se realiza en CPU. Por ello, la cantidad de parámetros del sistema debe ser tal que todos los parámetros puedan estar cargados en la RAM cuando se ejecuta el programa. Otra limitante que se tuvo en cuenta es el tiempo de inferencia, debido a que la inferencia en CPU lleva más tiempo que en GPU. Esto último se puede corroborar en el anexo B, donde se encuentran

las pruebas que se realizaron midiendo el tiempo de inferencia de los modelos en una computadora personal.

Estas pruebas junto a la cantidad de parámetros de los modelos fueron las principales características que se usaron para seleccionar modelos válidos para el uso de la empresa.

Se deja planteado como trabajo futuro la experimentación con modelos con una mayor cantidad de parámetros, los cuales quedaron excluidos en esta etapa debido a las limitaciones mencionadas. La investigación del estado del arte sugirió que estos podrían mejorar los resultados, aunque no de forma significativa.

A continuación se mencionan los modelos seleccionados según su familia:

Inception

De la familia *Inception* (3.2.5) se seleccionó el modelo Inception v3. Se decide experimentar con este modelo por tener un buen rendimiento relativo a su costo computacional al combinar varias convoluciones de diferentes tamaños y capas de agrupación.

ResNet

De la familia *Residual Networks* (3.2.1), se seleccionaron los modelos ResNet50, ResNet101 y ResNet152. Se optó por la utilización de estos modelos dado que fueron el estado del arte en clasificación de imágenes hasta la introducción del Vision Transformer (3.2.3). Se considera que esta familia ofrece buenos resultados para el problema general, sin introducir mayor complejidad en el entrenamiento.

DenseNet

De la familia *Densely Connected Convolutional Networks* (3.2.2) se seleccionaron los modelos DenseNet121, DenseNet161 y DenseNet201. Proporcionan un enfoque alternativo, fortaleciendo el flujo de información de forma diferente a ResNet teniendo en cuenta el dominio específico en cuestión.

ViT

De la familia *Vision Transformer* (3.2.3), se seleccionó el modelo ViT-base. Este modelo fue seleccionado debido a que actualmente el estado del arte en clasificación de imágenes está dominado por modelos de la familia vision transformer, distinguidos por contener una gran cantidad de parámetros. Si bien las limitaciones planteadas anteriormente condicionan la cantidad de parámetros a utilizar, se consideró pertinente experimentar con modelos de esta familia.

DaViT

De la familia *Dual Attention Vision Transformer* (3.2.4) se seleccionó el modelo DaViT-base, siendo este, el modelo más nuevo al momento de la inves-

tigación. Mejora los resultados del vision transformer sin aumentar significativamente la cantidad de parámetros.

Para todos los modelos seleccionados, se encuentran disponibles sus pesos pre-entrenados en conjuntos de datos de clasificación de imágenes genéricos, los cuales están compuestos por imágenes de varios tipos. El equipo eligió únicamente modelos cuyos pesos estuvieran disponibles para poder trabajar con ellos aplicando *fine-tuning*.

4.2.3. Implementación

Para trabajar con los modelos seleccionados se construyó un programa el cual se denomina *trainer*. Este programa permite entrenar a los modelos seleccionados con los conjuntos de datos generados mediante la técnica de *fine-tuning*. El programa fue implementado en el lenguaje *Python* utilizando la biblioteca de aprendizaje automático *PyTorch*(*PyTorch*, s.f.).

Este programa permite sistematizar la experimentación con los distintos modelos seleccionados. Es pertinente mencionar que el entrenamiento de estos modelos es un proceso computacionalmente costoso que puede requerir horas o días de procesamiento, incluso cuando se dispone de hardware especializado. El programa *trainer* permite pivotar rápidamente entre experimentos, pudiendo comenzar la experimentación entrenando con un conjunto de datos de pocas imágenes e incrementar el tamaño de las pruebas si se considera necesario. Más adelante, en la sección 4.3.3 se describe en detalle la metodología utilizada para experimentar.

El programa *trainer* tiene una interfaz de línea de comandos que expone mediante parámetros opciones para configurar el entrenamiento del modelo. La implementación y documentación del *trainer* se encuentra disponible en (*Proyecto de Grado - Redes neuronales para detección de defectos en saneamiento*, 2024).

Cargado de los modelos

Al empezar su ejecución, el programa *trainer* carga los pesos del modelo seleccionado, pre-entrenado en el conjunto de datos *ImageNet-1K* (Deng y cols., 2009). Este es un conjunto de datos que contiene imágenes de 1.000 categorías distintas, el cual es ampliamente utilizado para investigación en el área de visión computacional. Se tomó esta decisión por dos motivos: el primero es que se considera que el conjunto de datos *ImageNet-1K* permite a la red pre-entrenada aprender a detectar características generales en imágenes que son aplicables también para el dominio específico tratado, que es lo que se busca al realizar *fine-tuning*. El segundo, es que existe una colección de modelos ya entrenados con *ImageNet-1K* que se encuentran disponibles a través del módulo *Torchvision* de *PyTorch* y de la biblioteca *timm* (*timm*, s.f.). Esto permite incorporar nuevos modelos con facilidad al *trainer*.

Por tanto, el programa obtiene el modelo que se quiere entrenar por medio de un parámetro, y en base a ello, carga sus pesos ya entrenados utilizando la

biblioteca correspondiente.

Entrenamiento

Para entrenar nuevos modelos se implementa un bucle de entrenamiento, el cual permite variar, mediante parámetros, distintos aspectos del mismo.

Como se mencionó, en la sección 4.2.1, los conjuntos de datos generados se separan en tres subconjuntos: entrenamiento, validación y prueba. Para el entrenamiento se utilizan los primeros dos subconjuntos.

En el anexo A se describe el funcionamiento del bucle de entrenamiento de los modelos.

4.3. Experimentación

La mayor parte del trabajo realizado en este proyecto, consistió en la experimentación con distintos modelos de clasificación de imágenes. Puntualmente, se entrenaron los modelos seleccionados con los conjuntos de imágenes de saneamiento que fueron generados.

Para la experimentación, se utilizó el programa *trainer*, cuya implementación se describe en la sección 4.2.3. Este programa, permite variar los diferentes parámetros del entrenamiento, así como elegir distintos modelos y conjuntos de datos para entrenarlos.

A continuación, se describe la infraestructura empleada, las métricas seleccionadas para la evaluación de los modelos, y la metodología seguida, en el proceso de experimentación. Finalmente, se presentan los mejores modelos obtenidos.

4.3.1. Infraestructura

El entrenamiento de modelos neuronales como los utilizados en este trabajo, radica en un proceso computacionalmente intensivo que requiere de hardware especializado para lograr su ejecución en tiempos razonables. En este contexto, se utilizó la infraestructura de ClusterUY (Nesmachnow y Iturriaga, 2019) para llevar a cabo el entrenamiento de los modelos. Esta misma, ofrece hardware especializado y permite la instalación de las bibliotecas de software que fueron utilizadas en el *trainer*.

ClusterUY ofrece dos tipos de unidades de hardware que permiten acelerar el entrenamiento de redes neuronales en PyTorch: GPUs NVIDIA Tesla P100 y GPUs NVIDIA A100 Tensor Core. Debido a su mayor disponibilidad y menor demanda, las GPUs P100 fueron el recurso que más se utilizó para el proyecto.

El cluster ofrece un entorno con un sistema operativo Linux donde los usuarios pueden instalar software. El equipo utilizó la herramienta *pyenv* (*pyenv*, s.f.) para gestionar la versión de Python y las bibliotecas necesarias dentro del entorno del cluster. Esta elección facilitó la configuración y mantenimiento del entorno de desarrollo, asegurando la compatibilidad y reproducibilidad de los experimentos realizados.

Para el entrenamiento, se transfirieron los conjuntos de datos generados y el programa *trainer* al cluster. El cluster requiere que las tareas a ejecutar se presenten en un formato específico, ya que utiliza el software *slurm* (Yoo, Jette, y Grondona, 2003) para el manejo y planificación de la ejecución de los trabajos. El equipo creó scripts para la ejecución de los trabajos que especifican el hardware a utilizar, en particular las GPUs P100/A100 y la transferencia de los conjuntos de datos a un almacenamiento de alta velocidad previo al entrenamiento.

El entrenamiento de los modelos con esta configuración, tuvo una duración variable según el modelo entrenado y el tamaño del conjunto de datos seleccionado. Estas duraciones, se encontraron en el rango de 6 a 72 horas. Una limitación que posee la plataforma del cluster es el tiempo máximo que se puede ejecutar un trabajo (72 horas). Sin embargo, no consistió en un impedimento dado que para la mayoría de los modelos, el entrenamiento convergió antes de dicho tiempo. Igualmente, se intentó seleccionar el parámetro de máximo de épocas para que los trabajos terminaran antes de las 72 horas, y se agregó al programa *trainer* un temporizador que guarda el modelo entrenado si se llega al limite de tiempo.

4.3.2. Métricas

La capacidad de evaluar los modelos es necesaria para permitir variar los experimentos realizados y mejorar los resultados obtenidos. Es por esto que para medir y cuantificar el rendimiento de los modelos generados, se calcularon distintas métricas de los mismos, las cuales se presentan a continuación. En el anexo C se detallan en mayor profundidad.

- Exactitud: Indica el porcentaje de aciertos totales, tanto positivos como negativos. Se utiliza esta métrica ya que es un buen indicador del rendimiento general de un modelo.
- Precisión: Indica el porcentaje de aciertos en los casos en que el modelo clasifica una imagen como defectuosa.
- Exhaustividad: Indica el porcentaje de imágenes defectuosas que el modelo clasifica como no defectuosas.
- Valor F1: Métrica que toma en cuenta tanto la precisión como la exhaustividad y devuelve un valor que refleja a ambos.

Se decidió utilizar las métricas mencionadas luego de investigar cuáles eran las más utilizadas para problemas parecidos al de esta investigación, para luego elegir las que más se adapten al mismo. Para cada una de estas métricas, se efectuó el cálculo para cada clase de defecto, para posteriormente promediar estos valores y utilizarlos como punto de referencia de comparación entre modelos. En base a ello, se decide cuáles reportaron mejores resultados a efectos de la investigación.

La implementación del calculo de las métricas se encuentra en el programa *trainer*.

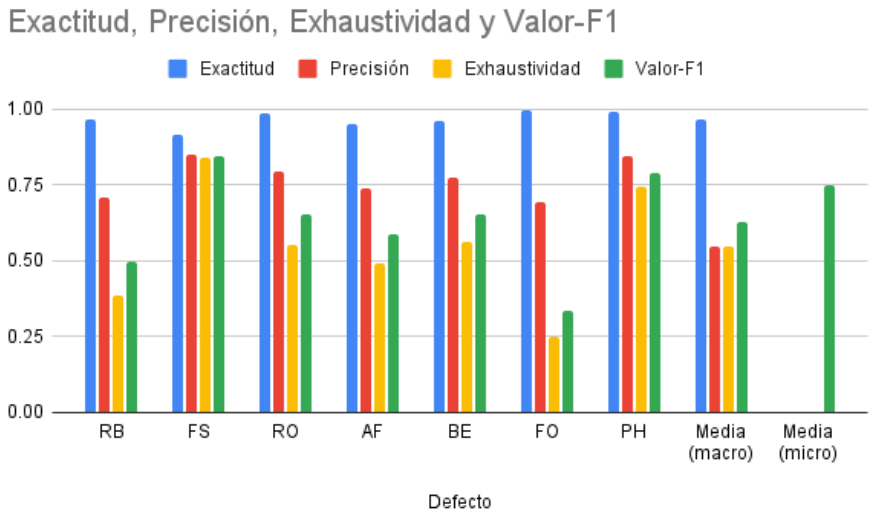


Figura 4.3: Gráfica de las métricas de ResNet152 entrenado con 1.170.000 imágenes

A modo de ejemplo, se presentan en la gráfica 4.3 los resultados de uno de los modelos que presentó mejores resultados, el ResNet152 entrenado con 1.170.000 imágenes. Para cada métrica de las mencionadas, se calcula dicha métrica para cada defecto y luego un promedio.

Como se puede observar en la gráfica 4.3 para cada defecto se tienen las métricas exactitud, precisión, y exhaustividad, y a su vez estas métricas para todos los defectos se promedian, lo cual da un valor que resume el rendimiento en esa métrica para todos los defectos. En la gráfica también se muestra la métrica F1, para esta métrica se calculan tanto el promedio macro como el micro (se explica más a detalle acerca de los valores micro y macro F1 en el anexo C).

Como se mencionó anteriormente y como se puede observar en la gráfica, se tiene que para el defecto RB (*Cracks, breaks, and collapses*) sucede que cada vez que el modelo predice ese defecto, se tiene un 70.6 % de probabilidades de acertar (precisión), mientras que se detecta el 38.6 % de los casos donde existe ese defecto (exhaustividad). A su vez, se tiene que para cada imagen, se acierta su presencia o ausencia en un 96.6 % de los casos (exactitud).

4.3.3. Metodología

El universo de experimentos que es posible realizar variando los parámetros del programa *trainer*, los modelos seleccionados y los conjuntos de datos generados, es considerablemente grande. Esto último sumado al tiempo que dura el

proceso de entrenar un nuevo modelo, dificulta realizar experimentos para cada modelo posible. Es necesario hacer una selección de los modelos a entrenar, y para ello se debe definir una metodología a seguir.

Se decidió seguir un enfoque incremental, comenzando la experimentación entrenando modelos más pequeños con menor cantidad de imágenes. Esto permitió empezar a experimentar probando distintas configuraciones, ya que estos experimentos requieren de un menor tiempo que los experimentos con modelos más grandes y mayor cantidad de imágenes.

Se divide el entrenamiento en tres etapas, una por cada conjunto de imágenes generado.

Etapa inicial

En la etapa inicial, se entrenó con el conjunto de datos más pequeño que contiene 120.000 imágenes. Se entrenó al menos un modelo de cada familia, empezando por los más pequeños para las familias con varios modelos, y se evaluó su desempeño en las métricas elegidas. En esta etapa, se experimentó variando algunos parámetros del entrenamiento como la tasa de aprendizaje y se probó variar el optimizador y la función de pérdida.

Asimismo, se experimentó generando modelos en dos etapas, donde primero las imágenes pasan por un clasificador binario que clasifica a la imagen como defectuosa o no defectuosa, y luego en caso de ser defectuosas pasan por un clasificador multi-etiqueta que clasifica a las imágenes según los defectos que contienen. La experimentación con este tipo de modelos fue motivada por la literatura consultada (3.2.6), donde estos modelos mostraron buenos resultados. Sin embargo, en los experimentos realizados no se logró reproducir estos resultados. Por este motivo sumado a que los clasificadores en dos etapas poseen la desventaja de requerir hacer inferencia dos veces, con el tiempo que eso conlleva, se descartó este enfoque para las siguientes etapas.

Etapa intermedia

Continuando con la experimentación incremental, en la etapa intermedia se entrenó con el conjunto de datos de 500.000 imágenes. Se realizó una selección de los mejores modelos de la etapa anterior según las métricas obtenidas, y se continuó la experimentación con estos mismos. Para las familias de modelos, se experimentó, de igual forma, con modelos más grandes. Por ejemplo, para la familia ResNet en la primera etapa, se experimentó con los modelos ResNet50 y ResNet101, mientras que en esta, se experimentó con ResNet152, un modelo más grande en términos de parámetros y capas.

Etapa final

Por último, la experimentación se traslada a una etapa final donde se reitera el proceso de selección de los mejores modelos a partir de la etapa anterior. En esta etapa, se entrenaron los modelos más grandes con el conjunto de datos de

1.170.000 imágenes. Se obtuvieron los mejores modelos en cuanto a los valores de las métricas, lo cual era esperable dado que se sabe que los modelos neuronales logran aprender a generalizar mejor cuando son entrenados sobre grandes volúmenes de datos.

En la siguiente sección se presentan los mejores modelos en términos de resultados obtenidos.

4.3.4. Resultados

Observaciones generales

De modo general, se consiguieron mejores resultados a medida que se fue acrecentando el conjunto de entrenamiento, esto se puede observar en las tablas del anexo D. A modo de ejemplo, para ResNet101 se tienen los resultados mostrados en la tabla 4.1, donde se puede observar la clara mejora de las métricas.

Imágenes	Épocas	Exactitud	Precisión	Exhaustividad	Macro F1	Micro F1
120.000	8*	0.9585	0.6305	0.5003	0.5532	0.6931
500.000	9*	0.9649	0.7331	0.5225	0.5997	0.7367
1.170.000	12*	0.9693	0.7789	0.5815	0.6587	0.7692

Tabla 4.1: Métricas para el modelo ResNet101 para los distintos tamaños de conjuntos de entrenamiento.

Se observa también que en general los modelos más grandes tienden a dar mejores resultados. Esto se puede apreciar también en las tablas del anexo D, donde los modelos con más parámetros tienden a tener mejores resultados, y sobre todo se puede apreciar en la comparativa dentro las distintas familias de modelos, donde las versiones más grandes suelen dar mejores resultados. Por ejemplo, en la familia DenseNet, ocurre que DenseNet-161 (29M de parámetros) tiene en general mejores métricas que DenseNet201 (20M de parámetros), el cual a su vez tiene en general mejores métricas que DenseNet121 (8M de parámetros).

En cuanto a los modelos basados en transformadores, no se observa una mejora significativa de las métricas con respecto a los modelos que utilizan cálculos convolucionales. Además, se puede apreciar dentro de los modelos basados en transformadores que DaVit dio mejores resultados que Vit, tal como se puede ver en la tabla 4.2.

Imágenes	Modelo	Épocas	Exactitud	Precisión	Exhaustividad	Macro F1	Micro F1
120.000	ViT-base	3	0.9570	0.7695	0.3646	0.4457	0.6720
	DaViT-base	8*	0.9592	0.6856	0.4654	0.5393	0.6836
500.000	ViT-base	11*	0.9645	0.7730	0.4959	0.5900	0.7274
	DaViT-base	8*	0.9653	0.8113	0.4931	0.6013	0.7156
1.170.000	ViT-base	8*	0.9626	0.8030	0.4293	0.5433	0.6876
	DaViT-base	9*	0.9702	0.7919	0.6050	0.6794	0.7790

Tabla 4.2: Resultados de los entrenamientos de ViT-base y DaViT-base. Datos extraídos del anexo D.

Mejores modelos

Siguiendo la metodología presentada en la sección anterior, el equipo realizó una serie de experimentos que tuvieron como resultado la generación de varios modelos sobre los que se obtuvieron métricas para evaluar su rendimiento. Los resultados experimentales se encuentran en el anexo D.

A partir de dichos resultados, se efectuó una selección de los mejores modelos, los cuales fueron elegidos no solo en base al rendimiento según las métricas, sino que también se tuvo en cuenta el tiempo de inferencia de cada uno.

A continuación, se comentan los modelos que fueron elegidos como mejores modelos y la motivación detrás de su elección.

- **DaVit-Base**

  Este es el modelo que obtuvo mejores resultados en los experimentos realizados: en todas las métricas evaluadas supera a los demás modelos. Es también el más nuevo con el que se experimentó.

  Tiene la desventaja de ser un modelo considerablemente grande, ya que cuenta con 88 millones de parámetros. Es también el modelo cuya inferencia exige un mayor tiempo de cómputo, como se puede apreciar en la tabla E.1.

- **ResNet152**

  Este es un modelo que obtuvo resultados ligeramente inferiores a DaViT-base, pero tuvo la ventaja de tener menos parámetros y menor tiempo de inferencia. Por este motivo, se utilizó en gran medida para trabajar con la generación automática de reportes.

  La arquitectura ResNet fue estado del arte para clasificación de imágenes hasta hace muy poco, y en estos experimentos se mostraron muy buenos resultados en relación a la cantidad de parámetros y tiempo de inferencia.

- **ResNet101 y ResNet50**

  Estos dos modelos obtuvieron resultados inferiores en las métricas respecto a los obtenidos para los modelos anteriores. Sin embargo, poseen un tiempo de inferencia considerablemente menor a ViT-base y ResNet152.

De los modelos con un tiempo de inferencia similar que fueron utilizados en la experimentación, estos fueron los que reportaron un mejor desempeño. Por este motivo, se seleccionaron considerando su uso en aplicaciones con restricciones de tiempo, donde se puede sacrificar rendimiento en métricas para ganar velocidad en la inferencia (para más información consultar el anexo E). Un ejemplo de esto son las aplicaciones en tiempo real. Llevado al caso de estudio, podría requerirse simultáneamente realizar la inferencia y la inspección de una tubería por parte de un operador, de manera de que este obtenga retroalimentación en vivo.

Modelo	**Parámetros**	**Lote**	**Exactitud**	**Precisión**	**Exhaustividad**	**Macro F1**	**Micro F1**
ResNet-50	26M	64	0.9685	0.7575	0.5690	0.6435	0.7628
ResNet-101	45M	64	0.9693	0.7789	0.5815	0.6587	0.7692
ResNet-152	60M	64	0.9690	0.7627	0.6025	0.6677	0.7707
ViT-base	86M	64	0.9626	**0.8030**	0.4293	0.5433	0.6876
DaViT-base	88M	128	**0.9702**	0.7919	**0.6050**	**0.6794**	**0.7790**

Tabla 4.3: Resultados de los mejores modelos al ser entrenados con 1.170.000 imágenes. Datos extraídos del anexo D.

Es pertinente mencionar que a la hora de elegir un modelo para una aplicación práctica se debe tener en cuenta varios factores, como el hardware disponible y las restricciones de tiempo. En general, los modelos que reportaron mejores métricas son también los que más demoran en la inferencia.

En la tabla 4.3 se muestran las métricas obtenidas para los modelos seleccionados.

4.4. Desarrollo de la solución

Como se mencionó anteriormente, una de las etapas del proyecto se destinó al desarrollo de un sistema para la empresa DICA & Asociados, que permite la automatización de la tarea de clasificación de defectos sobre videos de inspección mediante el empleo de modelos neuronales.

Para asegurar que el producto final provea valor al cliente, se establecieron distintos requerimientos en conjunto con representantes de la empresa DICA & Asociados.

Actualmente, la dinámica operativa de la empresa radica en la inspección de tuberías utilizando dispositivos robóticos (ver figura 4.4), que son introducidos en las tuberías y operados desde la superficie (ver figura 4.5), para la recolección de videos que posteriormente son analizados por profesionales en el área. Estos profesionales son los responsables de la clasificación de los distintos defectos encontrados en la revisión de un video. Dicha tarea, se lleva a cabo de manera manual, donde el especialista técnico, metodológicamente, reporta cada defecto

hallado a medida que inspecciona el video recolectado. Como resultado final, el personal elabora un informe que incluye una descripción de aquellos defectos identificados e imágenes adjuntas de cada uno.

El sistema que se propone pretende agregar valor a su labor actual y para ello proporciona dos modos de uso; el primero corresponde a la generación automatizada de reportes de defectos, donde la clasificación es efectuada por modelos neuronales sobre un video-inspección de entrada, y el segundo, a la generación de un video que exhibe de manera gráfica el proceso de clasificación del modelo sobre dicho video de entrada.

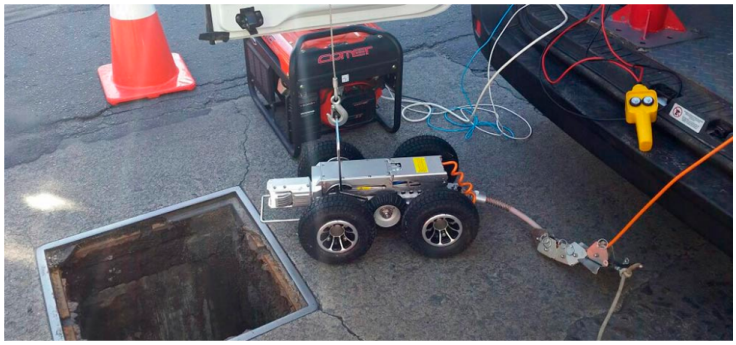


Figura 4.4: Robot próximo al ingreso en la tubería. Figura extraída de (Asociados, s.f.).

Figura 4.5: Técnico operando el dispositivo robótico desde la superficie. Figura extraída de (Asociados, s.f.).

4.4.1. Diseño

De modo de llevar a cabo la automatización del proceso de análisis de videos y la clasificación de defectos, el sistema debe recibir un video-inspección como entrada y generar como salida un documento con los defectos encontrados, y

opcionalmente, un video que exhiba gráficamente el proceso de clasificación. Para dicho propósito, se emplean los modelos entrenados que obtuvieron mejores resultados, presentados en la sección 4.3.4. En este sentido, se diseñaron tres módulos responsables de la ejecución de cada tarea; uno dedicado a la clasificación de defectos, otro, a la generación del documento o informe de los defectos hallados, y por último, uno destinado a la generación del video. Esta estructuración, responde a la intención de disponer de ambientes aislados y consistencia en el desarrollo de cada tarea y del sistema en general. Finalmente, se tiene un programa principal que interactúa con cada módulo de modo de unificar la comunicación con cada uno y servir de intermediario con el usuario final, quien a su vez interactúa con dicho programa a través de una interfaz gráfica. Esta interfaz permite de manera visual e intuitiva la generación de reportes de defectos y videos de clasificación, pudiendo asimismo, configurar ciertos parámetros de cada funcionalidad.

A continuación se describen en mayor profundidad el funcionamiento de cada módulo y el programa principal.

Módulo de clasificación de defectos

Este módulo, como su nombre lo indica, es responsable de ejecutar el proceso de clasificación de defectos sobre un video de inspección de entrada. Para este proceso, se emplean los modelos entrenados que consiguieron mejores resultados en la parte experimental del proyecto. El usuario puede seleccionar cuál utilizar, y por consiguiente, el módulo cargará los pesos del modelo elegido para efectuar la clasificación. Los modelos seleccionados se comentan en la sección 4.3.4. Los distintos tipos de defectos a clasificar son los especificados en 4.2.1.

Asimismo, el módulo carga el video de entrada y procede a realizar un proceso de inferencia sobre este, utilizando el modelo seleccionado. Dado que los modelos fueron entrenados para la tarea de clasificación sobre imágenes, se aplica el modelo sobre cada cuadro (o *frame*) del video. Luego, para cada cuadro o imagen se almacenan en una estructura de datos las predicciones del modelo para cada defecto, donde su posición en la estructura indica, implícitamente, el número de cuadro asociado (que será útil más adelante). Estas predicciones, son requeridas por los módulos de generación de documento y de video para llevar a cabo sus procesamientos.

Módulo de generación de documento

En base a las predicciones generadas por el módulo de clasificación y un documento plantilla (*template*) de entrada, genera un nuevo documento que contiene una descripción para cada defecto hallado en el video-inspección e imágenes de cada uno. Dicha plantilla, representa la estructura básica que la empresa emplea actualmente para realizar sus reportes y entregárselos a sus clientes. Por tanto, el módulo completa la plantilla con la información recolectada del video-inspección y genera un nuevo documento de salida o reporte.

Tal como se mencionó, este módulo invoca al módulo de clasificación para obtener las predicciones del video de entrada, que según se describió, se almacenan en una estructura de datos. Esta estructura posee, para cada cuadro del video, las predicciones para los distintos tipos de defectos. Para la clasificación de defectos, se consideran únicamente aquellos cuadros donde existen defectos con probabilidades mayores a cierto umbral establecido. No obstante, dada la enorme cantidad de cuadros que comprende un video, existirían una vasta cantidad de imágenes (del orden de las miles) que cumplan dicha condición, resultando inviable reportar todos esos defectos e imágenes en el documento generado. Esto puede explicarse en que dos cuadros continuos suelen tener probabilidades semejantes para un mismo tipo de defecto, dado que ambos pertenecen a un mismo instante en el tiempo. En otras palabras, en un video-inspección existen instantes donde se visualiza durante varios segundos un mismo defecto hasta que la vista se desplaza hacia otro plano de la tubería. Es esperable que durante esos segundos las predicciones de los cuadros involucrados indiquen la presencia de un defecto, y de hecho, del mismo defecto.

En la figura 4.6 se observa una gráfica del comportamiento de un modelo sobre un período de diez segundos de un video-inspección dado. Cada línea representa un tipo de defecto indicado por su color. Si se asume un umbral del 50 % (línea punteada negra en el eje y), se aprecia que el defecto "Rotura/fisura" (en rojo) se encuentra por encima de dicho valor durante varios segundos en el tiempo. Si a estos se los multiplica por los 25 fps que tiene el video, se tiene una gran cantidad de cuadros que refieren al mismo defecto, siendo superfluo incluirlos todos en el documento a generar. Basta con seleccionar un único cuadro de todos ellos que represente la presencia de dicho defecto (se explica más a detalle en la sección 4.4.2).

Adicionalmente, se aprecia cómo las predicciones oscilan drásticamente entre ciertos cuadros, pudiendo inducir una clasificación errónea para un defecto dado. Por ejemplo, si en un momento del video un defecto supera el umbral durante un segundo y luego vuelve a caer por debajo de él, pero en otro instante, el mismo defecto supera el umbral por más de tres segundos, sucede que en el primer caso la probabilidad de que tal defecto esté presente es menor que en el segundo caso donde se perpetúa más en el tiempo.

Consecuentemente, es necesario dictaminar una estrategia elaborada que permita, ingeniosamente, la elección de ciertos defectos e imágenes a incluir en el documento. Se diseñó entonces, una función heurística que toma como insumo la estructura de datos de las predicciones, ejecuta un proceso de clasificación sobre estas, y devuelve los números de cuadros y sus defectos asociados que se incluirán en el informe. Esta función permite, a su vez, regular la forma en que se efectúa el proceso de clasificación a través de ciertos parámetros. Detalles sobre su funcionamiento se encuentran en la sección 4.4.2.

Obtenidos los números de cuadros y defectos de la función heurística, se procede a extraer del video de inspección las imágenes asociados a dichos números e incluirlas en el documento junto con las descripciones de los defectos, devolviéndolo como salida del módulo.

En la figura 4.7 se visualiza un ejemplo de un documento de salida con las

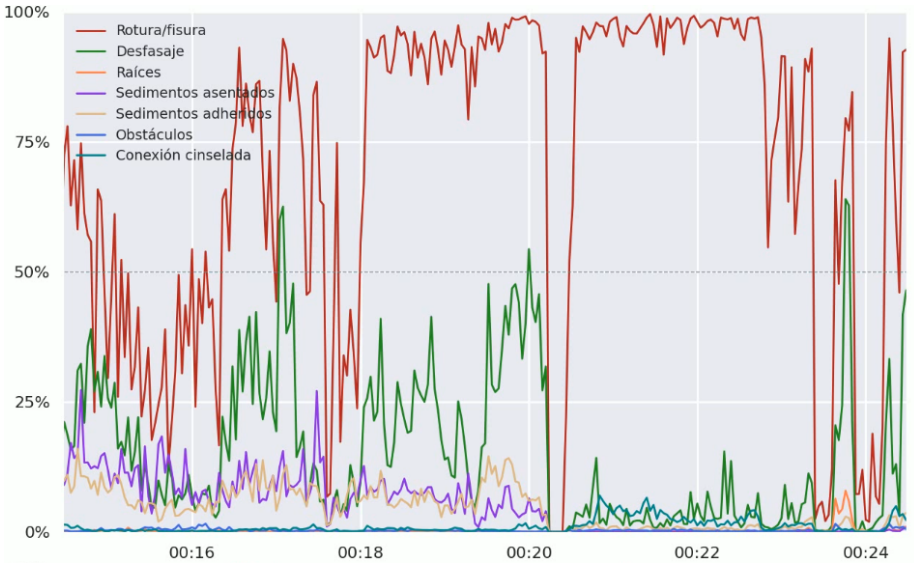


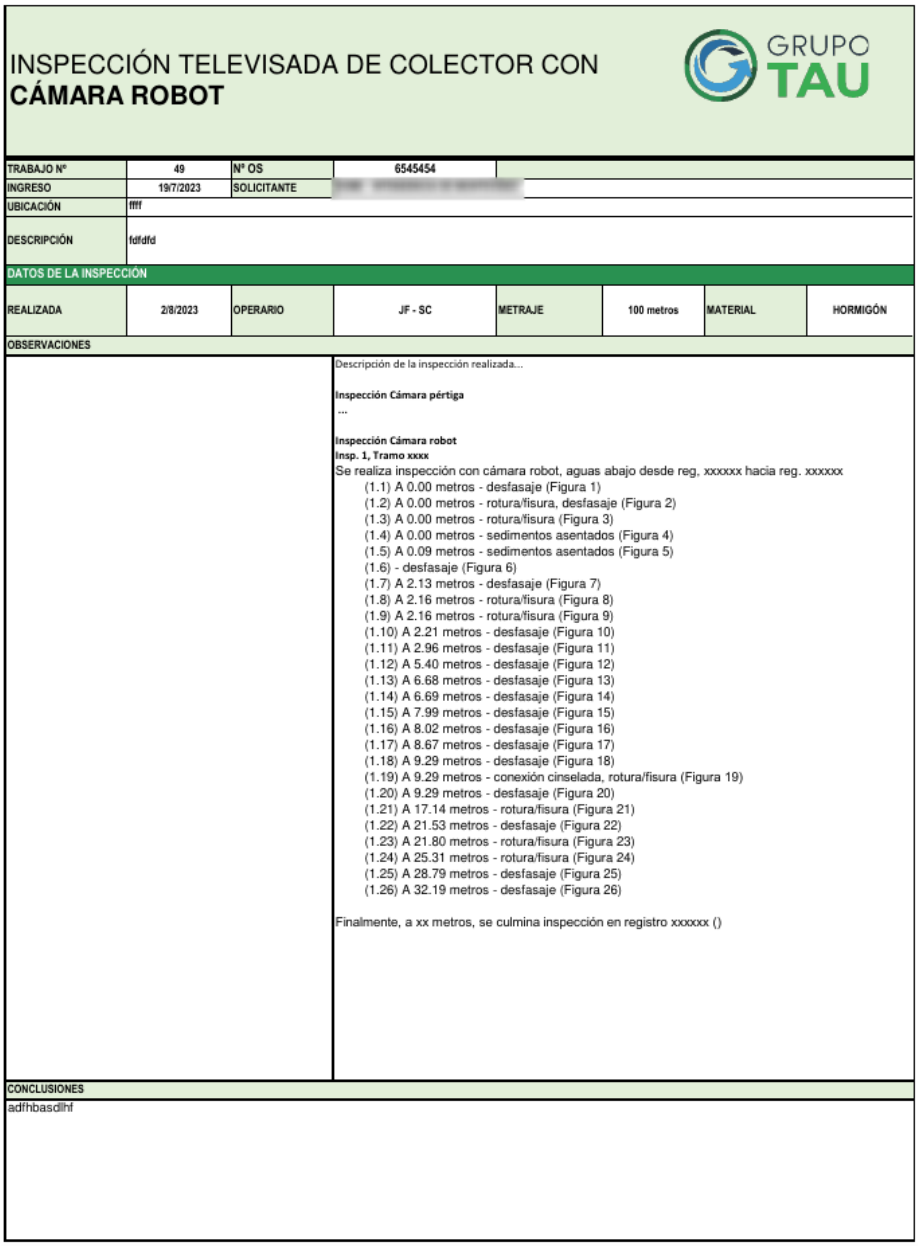
Figura 4.6: Variación de las predicciones de defectos, de un modelo sobre un video-inspección de 25 fps. El eje x representa el tiempo en segundos, y el eje y, el porcentaje de probabilidad. Cada línea representa la variación de un defecto indicado por su color. La línea punteada negra en el eje y marca un umbral del 50 %. Figura extraída de un video generado por el sistema.

descripciones de los defectos encontrados y la distancia a la que fueron identificados en la tubería. En la figura 4.8 se observa el mismo documento de salida, donde ahora se visualizan las imágenes adjuntas referentes a los seis primeros defectos vistos en la figura anterior.

Módulo de generación de video

Como se aludió con anterioridad, el presente módulo es el responsable de la generación del video que exhibe el proceso de clasificación de defectos sobre el video-inspección de entrada. Si bien esta utilidad no forma parte de la estructura actual de trabajo de la empresa, pretende complementar y aportar valor a la misma. El motivo de su desarrollo se encuentra en proveer a los operarios de la empresa contenido visual que exponga gráficamente el proceso de clasificación del modelo, particularmente, las variaciones en las predicciones. De esta manera, se puede identificar fácilmente los instantes del video de inspección que presentan una mayor concentración de defectos. Asimismo, esta funcionalidad enriquece el documento generado, pudiendo el personal recurrir a la misma en caso de demandar un mayor entendimiento o detalles del análisis de la inspección en cuestión.

El video consta de la integración de tres componentes. El primero es la repro-



INSPECCIÓN TELEVISADA DE COLECTOR CON **CÁMARA ROBOT**

GRUPO TAU

TRABAJO Nº	49	Nº OS	6545454				
INGRESO	19/7/2023	SOLICITANTE	[redacted]				
UBICACIÓN	ffff						
DESCRIPCIÓN	fdfdfd						
DATOS DE LA INSPECCIÓN							
REALIZADA	2/8/2023	OPERARIO	JF - SC	METRAJE	100 metros	MATERIAL	HORMIGÓN

OBSERVACIONES

Descripción de la inspección realizada...

Inspección Cámara pértiga
...

Inspección Cámara robot
Insp. 1, Tramo xxxx
Se realiza inspección con cámara robot, aguas abajo desde reg. xxxxxx hacia reg. xxxxxx
(1.1) A 0.00 metros - desfasaje (Figura 1)
(1.2) A 0.00 metros - rotura/fisura, desfasaje (Figura 2)
(1.3) A 0.00 metros - rotura/fisura (Figura 3)
(1.4) A 0.00 metros - sedimentos asentados (Figura 4)
(1.5) A 0.09 metros - sedimentos asentados (Figura 5)
(1.6) - desfasaje (Figura 6)
(1.7) A 2.13 metros - desfasaje (Figura 7)
(1.8) A 2.16 metros - rotura/fisura (Figura 8)
(1.9) A 2.16 metros - rotura/fisura (Figura 9)
(1.10) A 2.21 metros - desfasaje (Figura 10)
(1.11) A 2.96 metros - desfasaje (Figura 11)
(1.12) A 5.40 metros - desfasaje (Figura 12)
(1.13) A 6.68 metros - desfasaje (Figura 13)
(1.14) A 6.69 metros - desfasaje (Figura 14)
(1.15) A 7.99 metros - desfasaje (Figura 15)
(1.16) A 8.02 metros - desfasaje (Figura 16)
(1.17) A 8.67 metros - desfasaje (Figura 17)
(1.18) A 9.29 metros - desfasaje (Figura 18)
(1.19) A 9.29 metros - conexión cinselada, rotura/fisura (Figura 19)
(1.20) A 9.29 metros - desfasaje (Figura 20)
(1.21) A 17.14 metros - rotura/fisura (Figura 21)
(1.22) A 21.53 metros - desfasaje (Figura 22)
(1.23) A 21.80 metros - rotura/fisura (Figura 23)
(1.24) A 25.31 metros - rotura/fisura (Figura 24)
(1.25) A 28.79 metros - desfasaje (Figura 25)
(1.26) A 32.19 metros - desfasaje (Figura 26)

Finalmente, a xx metros, se culmina inspección en registro xxxxxx ()

CONCLUSIONES

adfhbasdlhf

Figura 4.7: Documento de salida del módulo, donde se describen los defectos encontrados en un video-inspección de entrada. Figura extraída de un documento generado por el sistema.

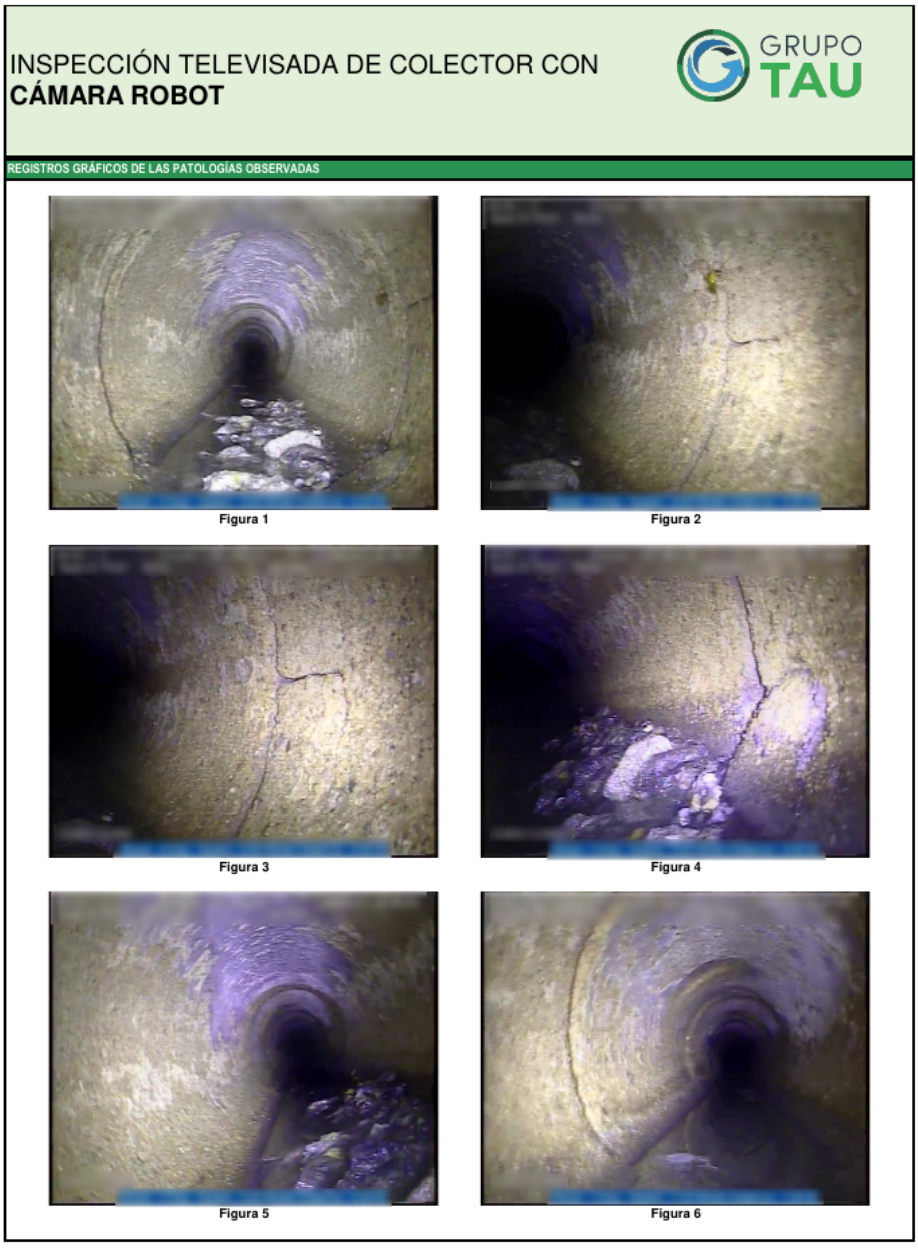


Figura 4.8: Documento de salida del módulo, donde se visualizan las imágenes correspondientes a los seis primeros defectos de la figura anterior. Figura extraída de un documento generado por el sistema.

ducción en miniatura del video-inspección de entrada. Luego hay dos gráficas, una estática y una dinámica.

La primera gráfica, capta y despliega un panorama general de las predicciones para cada defecto en todo el video-inspección. Esto permite a los operarios visualizar a simple vista el proceso de clasificación resumidamente y discernir los puntos en el tiempo con mayor presencia de defectos. En adición, la gráfica cuenta con un cursor que marca el punto que se está examinando en el video de inspección. En la figura 4.9 se observa una gráfica estática que resume el contenido de la clasificación para los seis primeros minutos del video de inspección. Cada línea representa las predicciones de un defecto en particular, y el cursor (línea vertical negra) dictamina el punto de examinación.

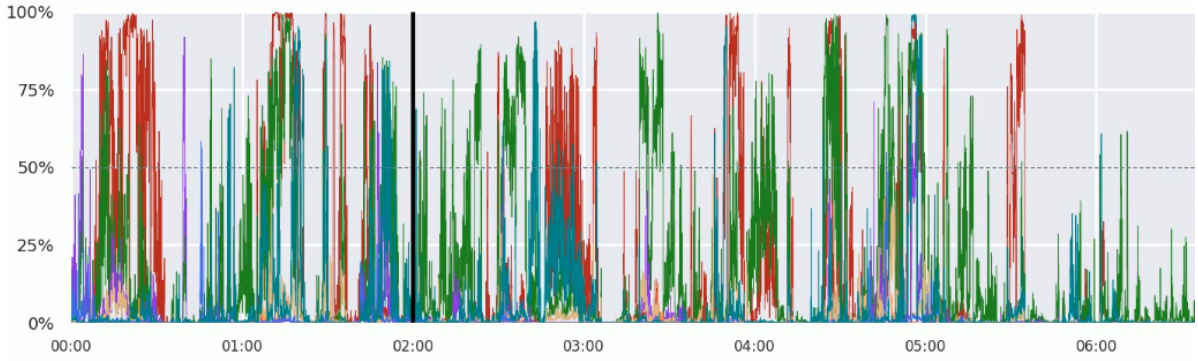


Figura 4.9: Gráfica estática para un video de inspección. El eje x representa el tiempo en minutos, y el eje y, el porcentaje de probabilidad. Cada línea representa la variación de un defecto indicado por su color. La línea vertical negra es el cursor que marca el punto que se examina en ampliación en la gráfica dinámica. Figura extraída de un video generado por el sistema.

Por otra parte, la gráfica dinámica actúa como ampliación del punto marcado por el cursor en la gráfica estática. La misma se va actualizando conforme avanza el video-inspección (de ahí su nombre) mostrando, como si fuese en tiempo real, el comportamiento de cada defecto en el proceso de clasificación. En la figura 4.10 se observa un ejemplo donde el instante de tiempo que se visibiliza es el mismo que el señalado por el cursor de la gráfica estática de la figura 4.9.

Para la construcción de ambas gráficas, el módulo hace uso de la estructura de datos con las predicciones de los defectos devuelto por el módulo de clasificación. En base a estos valores y el número de cuadro del video-inspección asociado, se construye el contenido de cada gráfica.

La idea de los tres componentes es que se permita visualizar en simultáneo las gráficas con los sucesos que acontecen en la tubería en análisis.

Recapitulando, el video generado consiste de la integración de los tres componentes, ambas gráficas y el reproductor del video. En la figura 4.11 se visualiza

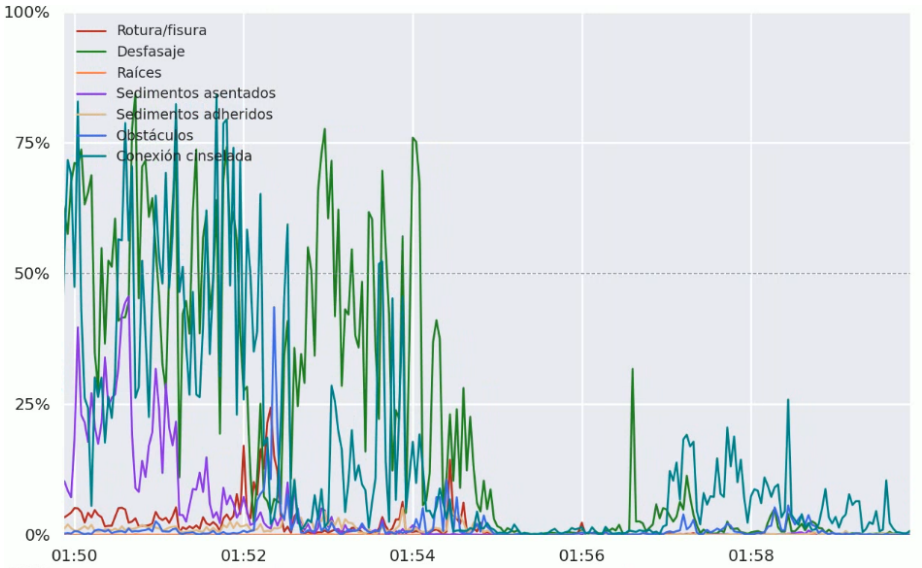


Figura 4.10: Gráfica dinámica para un video de inspección, que amplía el instante marcado por el cursor en la gráfica estática de la figura 4.9. El eje x representa el tiempo en segundos, y el eje y, el porcentaje de probabilidad. Cada línea representa la variación de un defecto indicado por su color. Figura extraída de un video generado por el sistema.

un instante de un video generado donde se observan los tres componentes mencionados, la gráfica dinámica (arriba a la izquierda), el reproductor del video de inspección (arriba a la derecha) y la gráfica estática (abajo).

Programa principal

Este programa pretende unificar la comunicación con los módulos descritos previamente, y actúa como punto de conexión con la interfaz de usuario. Asimismo, recibe todos los parámetros configurables del sistema pudiendo desde él ejecutar la generación de reportes de defectos hallados en una inspección, y la generación de videos de clasificación.

Este programa es el responsable de cargar el video-inspección especificado por el usuario, y acto seguido invocar al módulo de clasificación, quien carga los pesos del modelo seleccionado y efectúa la inferencia sobre el video, devolviendo una estructura con las predicciones de los defectos. El sistema ofrece la posibilidad de guardar dicha estructura en un archivo, de modo de poder cargar las predicciones del mismo video-inspección y generar nuevos reportes o videos variando distintos parámetros, ahorrándose el tiempo de cómputo que conlleva la inferencia. Los tiempos de ejecución de cada módulo se detallan en el anexo E.

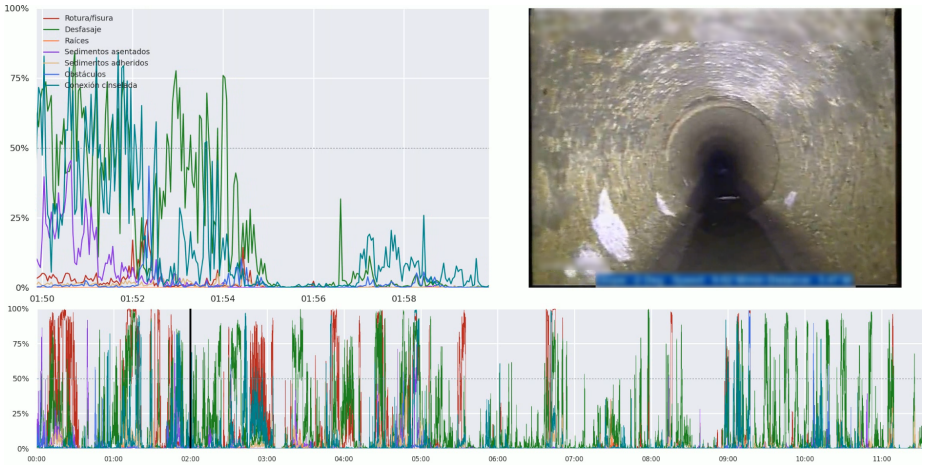


Figura 4.11: Video de clasificación generado para un video de inspección de entrada. Se observa la disposición de la gráfica dinámica (arriba a la izquierda), el reproductor en miniatura (arriba a la derecha), y la gráfica estática (abajo). Figura extraída de un video generado por el sistema.

Obtenidas las predicciones, se procede a la generación de reportes y videos. El sistema genera reportes de manera predeterminada, pudiendo el usuario determinar, adicionalmente, si desea la generación de un video de clasificación. En caso de la generación de reportes, el sistema carga el documento plantilla (*template*) y en base a las predicciones y el video-inspección, genera un nuevo reporte que se guarda en el sistema de archivos en la ruta especificada por el usuario. Análogamente, para la generación de videos.

En la figura 4.12 se visualiza el funcionamiento interno del sistema y cómo interactúan los distintos componentes.

Interfaz gráfica

El diseño de la interfaz constituye una pieza clave para que DICA & Asociados pueda interactuar fácil e intuitivamente con el sistema. Permite al usuario configurar los valores de los distintos parámetros por medio de elementos gráficos. En la figura 4.13 se observa la interfaz y sus componentes gráficos. Se puede apreciar las distintas opciones que posee la misma, particularmente, se destaca la parte de *Ajustes avanzados* (a la derecha en la figura) que permite la variación de parámetros relacionados a la generación de reportes. Se redactó un manual de usuario de modo que la empresa tenga a disposición un documento sobre el funcionamiento del sistema en caso de requerirlo. El mismo se encuentra en el repositorio de este proyecto (*Proyecto de Grado - Redes neuronales para detección de defectos en saneamiento*, 2024).

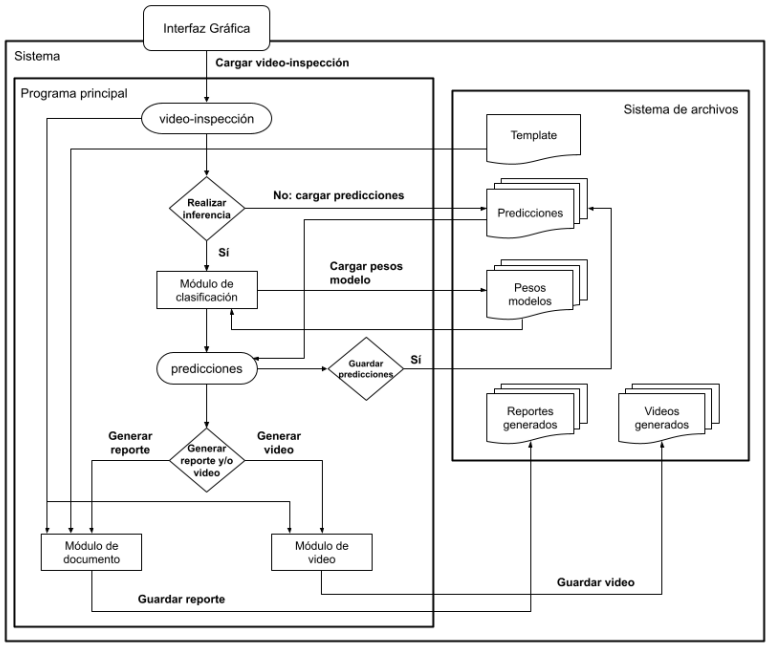


Figura 4.12: Diagrama de funcionamiento y componentes del sistema.

4.4.2. Implementación del sistema

La implementación de la lógica del sistema se realizó en el lenguaje *Python*. Para cada módulo se emplearon distintas bibliotecas que permitan el desarrollo de cada tarea. Para el módulo de clasificación se utilizó la biblioteca *PyTorch* (*PyTorch*, s.f.) para el proceso de carga de los modelos e inferencia sobre el video-inspección. Asimismo, se empleó la biblioteca *matplotlib* (*Matplotlib*, s.f.) para el módulo de generación de video para la construcción de las gráficas y el reproductor de video. En cuanto al módulo de generación de documento, se utilizó la biblioteca *openpyxl* (*OpenPyXL*, s.f.) para la manipulación y generación de documentos en formato *excel*, que corresponde al tipo de documento que utiliza la empresa para el reporte de los defectos hallados en una inspección. Por último, para la implementación de la interfaz se empleó la biblioteca *TKinter* (*TKinter*, s.f.) que permite la construcción de interfaces gráficas de usuario para el lenguaje *Python*.

Como se señaló en la sección de diseño 4.4.1, se implementó un programa principal que cumple la función de intermediario entre el usuario (interfaz) y los distintos módulos del sistema. Este programa, recibe los parámetros del sistema para la generación de reporte y video. Por tanto, es el usuario final, quien a

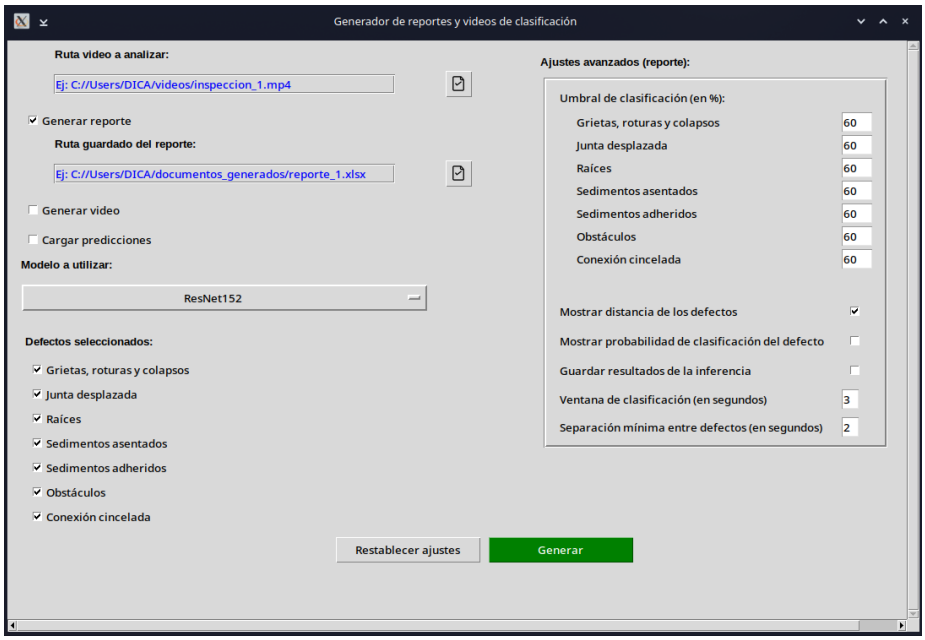


Figura 4.13: Interfaz gráfica del sistema.

través de la interfaz, podrá configurar los distintos parámetros con los valores que desee. Considerando que este programa es el responsable de la interacción con los distintos módulos, realiza algunas optimizaciones. Por ejemplo, dado que el proceso de clasificación de defectos se debe efectuar tanto para la generación del documento como para la del video, el programa invoca una única vez al módulo de clasificación y brinda la opción de guardar la estructura de datos con las predicciones en un archivo de extensión *.json*. Este flujo de funcionamiento se visualiza en el diagrama de la figura 4.12. Esta decisión se fundamenta en que la ejecución del proceso de inferencia de un modelo sobre un video puede tomar varios minutos en concretarse, con lo cual almacenando los resultados se evita volver a ejecutar dicho proceso. Resulta especialmente útil en caso de querer ajustar ciertos parámetros de la generación del documento (particularmente de la función heurística), basta con cargar el archivo con las predicciones y generar el documento nuevamente. Análogamente, en caso de querer generar un video.

Según lo estipulado con la empresa, el funcionamiento por defecto del sistema es la generación de documentos de defectos, pudiendo opcionalmente generar videos de clasificación si el usuario así lo desea. El proceso de generación de documentos requiere un tiempo de ejecución significativamente menor con respecto a la generación de videos. En el anexo E se exponen los tiempos de ejecución de cada tarea, especificando el hardware donde fue ejecutada.

Como se mencionó, la generación de reportes no solo requiere de las predicciones del módulo de clasificación, sino que también de la aplicación de una función heurística, que en base a dichas predicciones, realiza un proceso de filtrado devolviendo los defectos e imágenes a incluir en el documento. Su creación responde a la problemática que entraña el hecho de manipular grandes números de cuadros en un video-inspección, donde varios de estos refieren a un mismo defecto. En la sección 4.4.1 se amplía acerca de esta cuestión. Por tanto, esta función constituye y desempeña un papel primordial en el funcionamiento del sistema. A continuación se detalla su implementación.

Función heurística

Esta función devuelve una estructura de datos que contiene los números de cuadros del video-inspección. Para cada cuadro, se tiene una lista de todos los defectos presentes en el mismo, y para cada defecto, la probabilidad asociada y el nombre. Luego, utilizando el número de cuadro, se procede a extraerlo del video de inspección de entrada e incluirlo como imagen en el documento a generar junto con la descripción de los defectos presentes.

Para la selección de los cuadros se debe formular una regla o estrategia que defina cuándo existe realmente un defecto en determinado instante del video-inspección. Observando el comportamiento de varios modelos en el proceso de inferencia sobre videos de inspección, se constató que cuando existe un defecto en un instante dado, la predicción del mismo suele superar un umbral de clasificación (por ejemplo, del 50 %) durante varios segundos. En la gráfica de la figura 4.6 se aprecia como el tipo de defecto "Rotura/fisura" (en rojo) permanece por encima del umbral (línea punteada horizontal) durante más de dos segundos (del segundo 18 al 20, y del 20 al 23 aproximadamente), y con valores de probabilidad entorno al 90 %. Si se considera además que el video es de 25fps, se deduce que el mismo defecto está presente en más de 50 cuadros consecutivos. Por tanto, la heurística debe ser capaz de identificar que todos esos cuadros corresponden al mismo defecto y así evitar incluirlos todos en el documento, resultando redundante a efectos de un análisis.

Se construyó una heurística que considera un intervalo o ventana de un número determinado de cuadros, que a su vez, se desplaza a razón de un cuadro sobre la estructura de las predicciones (que contiene todos los números de cuadros, y para cada uno, las predicciones de cada defecto). Para cada defecto, se realiza el promedio de las predicciones abarcadas por dicho intervalo, y si el valor obtenido supera un cierto umbral, se considera que el defecto está presente en el instante marcado por el intervalo. Se procede a almacenar en una nueva estructura dicho valor junto con el número de cuadro del intervalo que posea la probabilidad asociada más alta.

En la figura 4.14 se observa una representación gráfica de la estructura de las predicciones para un video de inspección de n cuadros (números debajo de la línea negra). Los y_{ij} refieren a los valores de las predicciones para el tipo de defecto i y el número de cuadro j. La heurística considera un intervalo de longitud l que se desplaza un cuadro en cada iteración. En la iteración t, el

intervalo rojo abarca las probabilidades encerradas por el recuadro del mismo color. Se calcula el promedio de las mismas, y en caso de superar el umbral de clasificación, se identifica el número de cuadro dentro del intervalo que posea la probabilidad asociada más alta, es decir, el cuadro $k = \{j|max(y_{ij}), j \in [x...x+l-1]\}$. Luego, se almacena el número de cuadro k junto con el valor del promedio en una nueva estructura, como se muestra en la parte inferior de la figura. En la siguiente iteración, $t+1$, se reitera el mismo procedimiento para las probabilidades abarcadas por el intervalo en azul.

El largo l de un intervalo se puede traducir a tiempo (en segundos) dividiendo l (número de cuadros) por los fps del video. Asimismo, es un parámetro del sistema que se puede ajustar según se quiera variar el tiempo que se considera la presencia de un defecto; intervalos más chicos aumentan la cantidad de defectos detectados mientras que intervalos de mayor longitud, la disminuyen. El umbral de clasificación también se puede especificar para cada defecto variando la sensibilidad del algoritmo en la clasificación de cada uno.

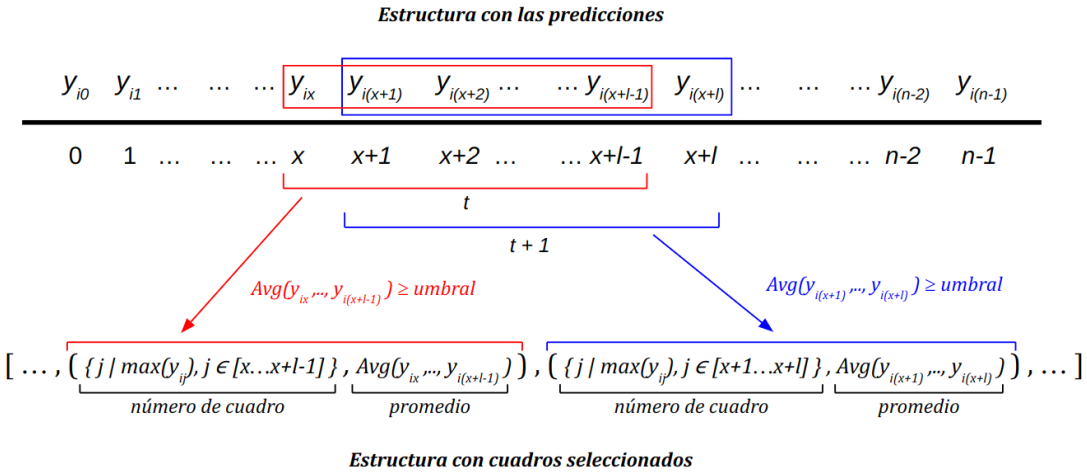


Figura 4.14: Representación gráfica de la estructura de datos con las predicciones (parte superior), para un video-inspección de n cuadros e intervalos de largo l. Los valores debajo de la línea negra corresponden a los números de cuadros. Por encima, los valores de las predicciones y_{ij}, donde i es un tipo de defecto y j el número de cuadro asociado. En la parte inferior, la estructura con los números de cuadros seleccionados y valores promedios, que se devuelve como salida.

Debido al solapamiento entre dos o más intervalos consecutivos, puede suceder que la probabilidad más alta para un mismo defecto, este asociada al mismo número de cuadro en iteraciones consecutivas. Para ello, la estructura contempla dicha situación almacenando una única vez el número de cuadro, pudiendo sobrescribir el valor del promedio asociado si este es mayor que el almacenado.

El proceso culmina una vez abarcados todos los cuadros del video por el intervalo, obteniendo como resultado una nueva estructura con los números de cuadros que representan la existencia de defectos en instantes determinados, y su probabilidades asociadas.

Esta estructura, a su vez, distingue los números de cuadros para cada tipo de defecto i de la figura 4.14 correspondiente a alguno de los mencionados en 4.2.1. Es decir, la estructura almacena, separadamente, los números de cuadros seleccionados según el tipo de defecto. Por otra parte, la estructura contendrá números de cuadros muy próximos que corresponden a un mismo instante de defecto, producto del desplazamiento del intervalo. Por ejemplo, si en la figura 4.14 el número de cuadro almacenado en la iteración t es $x + 1$, y en la $t + 1$ es $x + l$, entonces es altamente probable que ambos números de cuadros refieran a un mismo defecto debido a su cercanía. Por tanto, se procede a efectuar un proceso de unificación de datos sobre dicha estructura. Este proceso crea una colección de datos y recorre la estructura con los cuadros seleccionados; agrupa los números de cuadros cercanos en el tiempo, según cierta distancia d, elige el cuadro con probabilidad más alta de los agrupados, y lo agrega en la nueva colección. En la figura 4.15 se observa una representación gráfica del proceso de unificación de defectos. El proceso comienza con la estructura de cuadros seleccionados (parte superior) que contiene m parejas (c_i, z_i), donde c_i corresponde al número de cuadro y z_i a la probabilidad asociada, ambos valores representando el instante en el video-inspección de la ocurrencia del defecto y su probabilidad de veracidad. Esta estructura es el resultado que se obtuvo del proceso anterior (figura 4.14). Sobre esta misma, se procede a realizar agrupamientos de defectos según una distancia d entre los números de cuadros de cada pareja. En la figura, se observan intervalos de colores diferentes que representan agrupamientos distintos. Por tanto, aquellas parejas que cumplan con la particularidad de que su diferencia en número de cuadros es menor o igual que la distancia d, se las agrupa de manera conjunta en una nueva estructura, como se aprecia en la figura (parte intermedia). La idea subyacente es que los cuadros dentro de cada agrupamiento correspondan al mismo defecto en el video-inspección. La distancia d es un parámetro del sistema, y como tal, se puede ajustar. Obtenidos los agrupamientos, se procede a extraer un único cuadro que represente el mismo defecto, de manera de reducir la cantidad de cuadros. Para ello, se selecciona la pareja (c_i, z_i) cuyo z_i sea el máximo dentro de cada agrupamiento, obteniendo una nueva estructura con los defectos unificados, como se visualiza en la figura (parte inferior). De esta forma, no solo se reduce significativamente el tamaño de la estructura previa, sino que se simplifica la selección de cuadros. Se tiene un único cuadro que representa la presencia de un defecto en un instante dado, logrando reducir el número cuadros, y por tanto, de imágenes a incluir en el documento. Este proceso descrito, se realiza para la estructura de cuadros seleccionados de cada tipo de defecto.

Por último, una vez obtenidas las estructuras unificadas de cada tipo de defecto se procede a realizar el mismo proceso de unificación sobre todas estas

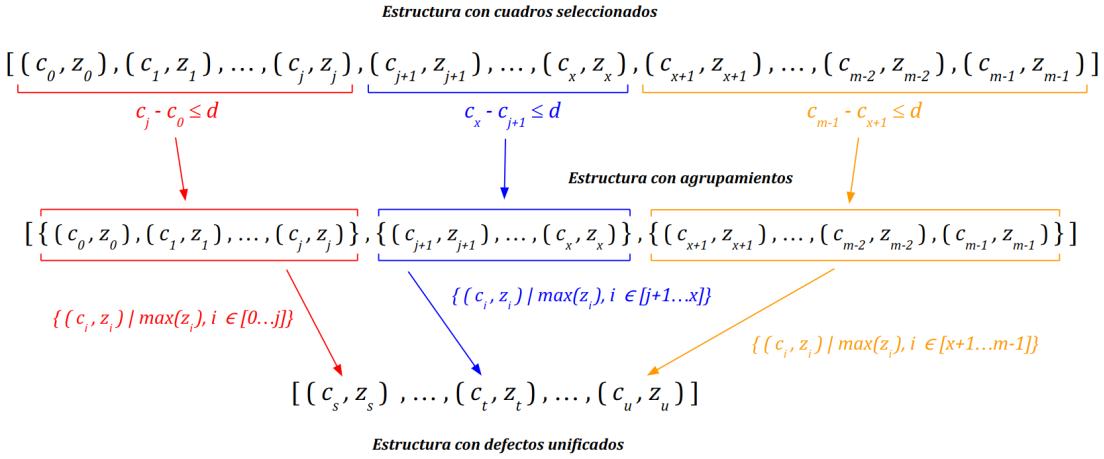


Figura 4.15: Representación del proceso de unificación de defectos. En la parte superior, se encuentra una estructura con los cuadros seleccionados para un tipo de defecto dado. Posee m parejas (c_i, z_i) donde c_i corresponde al número de cuadro y z_i a la probabilidad asociada. En la parte del medio, una nueva estructura que contiene los agrupamientos realizados sobre la estructura anterior, donde la diferencia entre los números de cuadros dentro de cada agrupamiento, es menor o igual que una distancia d. Por último, en la parte inferior, una nueva estructura con defectos unificados, que reduce significativamente el tamaño de las estructuras previas y simplifica la selección de cuadros.

estructuras, obteniendo así una única estructura final que será utilizada para la extracción de imágenes del video-inspección e incluirlas en el reporte a generar. En la figura 4.16 se observa una representación gráfica de dicho proceso. El proceso comienza con las estructuras con los defectos unificados (parte superior) de cada tipo de defecto, obtenidas del procedimiento anteriormente descrito. En la figura se aprecian estas estructuras marcadas en colores distintos, donde cada una corresponde a un tipo de defecto diferente. Asimismo, cada estructura contiene parejas (c_{ij}, z_{ij}), donde c_{ij} corresponde al número de cuadro y z_{ij} a la probabilidad asociada, siendo i el índice en la estructura y j el tipo de defecto. Notar que cada estructura posee distintas cantidad de cuadros seleccionados, por ejemplo la estructura en rojo tiene u cuadros seleccionados mientras que la azul, r cuadros. Continuando con el proceso, se colocan todos los elementos de las estructuras en una nueva colección (parte intermedia de la figura) ordenadas según el número de cuadro c_{ij}. Notar que ahora los números de cuadros pertenecen a tipos de defectos distintos, y además, sucede nuevamente que existen cuadros que se encuentran muy próximos a otros. Esto se debe a que en un mismo instante del video-inspección pueden ocurrir más de un defecto al mismo tiempo. Por tanto, se procede a efectuar un nuevo proceso de agrupamiento

de parejas cuadro-probabilidad según cierta distancia d. A diferencia del caso anterior, en lugar de tomar el cuadro con la probabilidad asociada más alta, se toma el primero de cada agrupamiento y se lo almacena en una nueva estructura (parte inferior de la figura) junto con una lista que contiene todos las probabilidades distinguidas según el tipo de defecto, contenidas en dicho agrupamiento. Notar que esta acción no se efectúa en el caso anterior por tratarse de cuadros de un mismo tipo de defecto. Finalmente, se obtiene una colección de datos que contiene únicamente los números de cuadros a extraer del video-inspección, y para cada uno, una lista con los tipos de defectos presentes y sus probabilidades. En el anexo F se encuentra un pseudocódigo sobre el funcionamiento de la función heurística.

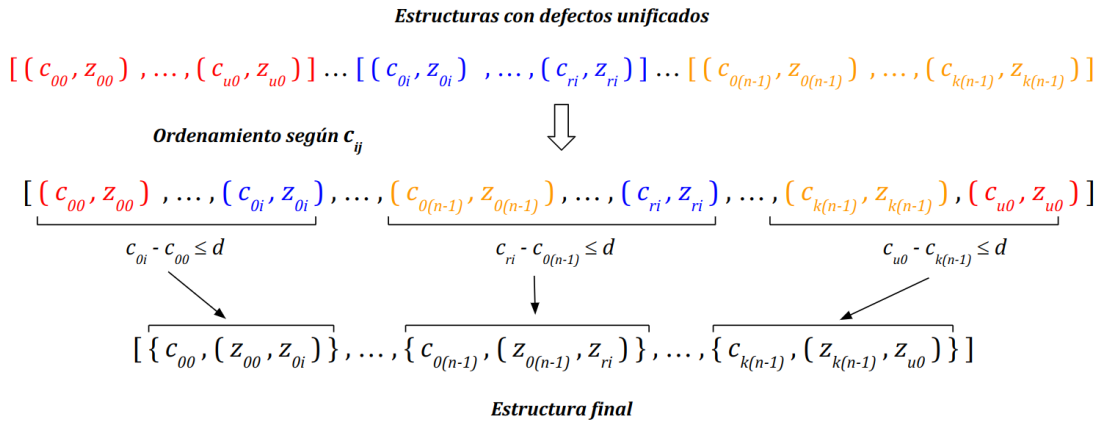


Figura 4.16: Representación del proceso de unificación final. En la parte superior, se encuentra una estructura con los cuadros seleccionados para cada tipo de defecto. Cada una, posee parejas (c_{ij}, z_{ij}) donde c_{ij} corresponde al número de cuadro y z_{ij} a la probabilidad asociada, siendo i el índice en la estructura y j el tipo de defecto. En la parte del medio, una nueva estructura que contiene las parejas de todas las estructuras anteriores, ordenadas según los c_{ij}. Por último, en la parte inferior, una nueva estructura con defectos unificados según los agrupamientos de la estructura anterior utilizando una distancia d entre los c_{ij}.

Extracción de distancia de los defectos

Se implementó la opción de incluir la distancia a la que se encuentra un defecto en la tubería, en el reporte generado. Los videos de inspección utilizados por la empresa son filmados con un dispositivo robótico que al grabar un video agrega información en la imagen sobre datos como la velocidad del robot y la distancia a la que se encuentra en la tubería. Por tanto, al momento de extraer un cuadro del video de inspección para incluirlo en el informe, se puede realizar

reconocimiento de texto y extraer el valor de distancia a la que se encuentra el robot, y por tanto, en donde fue detectado el defecto. Para ello, se utiliza la biblioteca *EasyOCR* de *Python* el cual consiste de un módulo de aprendizaje profundo que permite la identificación y extracción de texto de documentos. En la figura 4.7 se visualiza en la descripción de los defectos reportados, la distancia de cada uno.

Por defecto, el sistema tiene esta funcionalidad activada, pudiendo el usuario desactivarla en caso de que así lo desee.

4.4.3. Evaluación

El proceso de evaluación del sistema desarrollado no resulta una tarea sencilla ni directa, especialmente si se quiere conocer qué tan bien se desempeña el sistema en la generación de reportes. Notar que interesa la generación de reportes y no la de videos, por tratarse de la actividad principal que realiza la empresa. Además, la generación de videos es una funcionalidad complementaria a la de reportes, donde se visualiza el proceso de clasificación del modelo empleado sobre el video-inspección a analizar. Por lo cual, se estaría evaluando el modelo en sí, tarea que se presenta en la sección de experimentación 4.3.

Consiguientemente, se pretende evaluar la calidad de los reportes generados por el sistema, y para dicha tarea, se utiliza como referencia un documento realizado por DICA & Asociados para un video de inspección, y otro generado por el sistema, para el mismo video. Se comparan ambos documentos contando la cantidad de defectos reportados, y en base a las imágenes, determinar cuáles son el mismo.

A modo de ilustración, en la figura 4.17 se visualiza una página de un reporte generado por DICA & Asociados (izquierda), y otro generado por el sistema (derecha). Ambas páginas contienen seis imágenes de defectos pero solo tres de estas corresponden al mismo defecto en ambos reportes. La *figura 9* de ambos reportes son el mismo defecto, asimismo lo son la *figura 10* del reporte de la empresa y la *figura 11* del reporte del sistema, y por último, la *figura 11* de la empresa y la 12 del sistema. Este mismo procedimiento se efectúa para todas las páginas de ambos reportes evaluando la intersección de los defectos.

Particularmente, se generó un reporte con el sistema para un video de inspección de 11 minutos y 46 segundos de duración. El reporte generado por DICA & Asociados detectó un total de 16 defectos mientras que el generado por el sistema, 20. En cuanto a la intersección de defectos, fueron 11 los reportados por el sistema que coinciden con los de la empresa. Este dato, denota un buen desempeño en la detección de defectos por parte del sistema.

No obstante, el sistema tiende a detectar defectos que no se corresponden precisamente con los reportados por la empresa. Esto se debe a que el modelo seleccionado puede poseer cierta sensibilidad en la detección de un defecto en particular producto del conjunto de entrenamiento utilizado, contingencia que puede ser solventada mediante el ajuste de la sensibilidad en la detección para cada tipo de defecto. En el anexo G se encuentra una tabla que complementa la comparativa visual de ambos reportes.

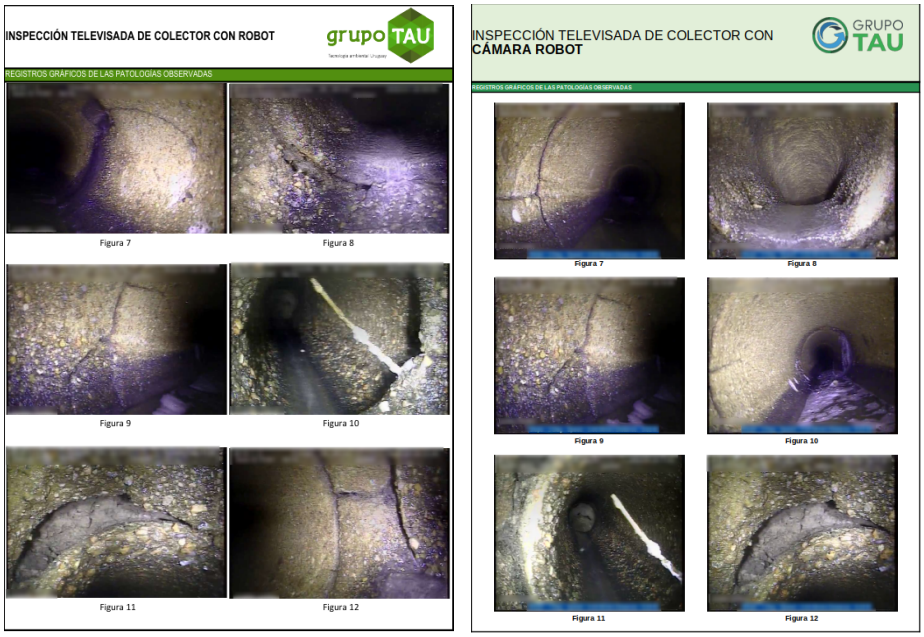


Figura 4.17: Reporte de defectos brindado por DICA & Asociados (izquierda), y generado por el sistema (derecha). La figura de la izquierda provista por DICA & Asociados.

Capítulo 5

Conclusiones y Trabajo Futuro

En este proyecto se trabajó en desarrollar redes neuronales para la detección de defectos estructurales en saneamiento a partir de un video. En este sentido, se consideraron los distintos enfoques posibles y se optó por usar redes de clasificación de imágenes. Fue viable entrenar estos modelos gracias a la disponibilidad del conjunto Sewer-ML (Haurum y Moeslund, 2021), el cual se utilizó para entrenar todas las redes generadas en el proyecto.

El mejor modelo generado fue DaViT-base entrenado con 1.170.000 imágenes de Sewer-ML. Este modelo obtuvo mejores resultados que los demás en todas las métricas, obteniendo una Micro F1 de 0.7790. Además, se generaron otros modelos que también brindan métricas cercanas a las de DaViT-base pero ofrecen tiempo de inferencia menores.

Un segundo objetivo del proyecto era generar un programa que permitiera a la empresa DICA & Asociados utilizar las redes neuronales generadas. En este sentido, el equipo mantuvo reuniones con personal de la empresa para evaluar sus necesidades y desarrollar un programa que permita automatizar la generación de reportes de video-inspecciones.

5.1. Conclusiones

Desde una etapa temprana del proyecto el equipo descubrió que la calidad del conjunto de datos usado impacta significativamente en los resultados de los modelos entrenados. La diferencia en la calidad de conjuntos de datos de defectos en saneamiento disponibles para detección de objetos y clasificación de imágenes fue lo que desató la decisión de enfocar la investigación exclusivamente en clasificación de imágenes. Puntualmente, el conjunto Sewer-ML dispone un gran volumen de imágenes clasificadas por profesionales. Se encontró que tener una gran cantidad de datos de calidad es un punto clave para que los modelos neuronales logren aprender a clasificar correctamente.

En relación a esto último, un punto importante y un riesgo que el equipo asumió fue que el conjunto de datos utilizado fue creado en un ambiente distinto en su naturaleza al ambiente con el que las redes generadas trabajarían. Esto es porque Sewer-ML fue creado con imágenes de video-inspecciones en Dinamarca, mientras que las redes generadas en este proyecto buscan ser utilizadas en el saneamiento de Montevideo. Tras probar las redes generadas con videos de inspecciones de la empresa DICA & Asociados, se encontró que las redes generadas funcionan correctamente para las imágenes del saneamiento local.

Respecto a los modelos con los que se experimentó, se encontró que en general dentro de una familia los modelos más grandes obtienen mejores resultados. Por ejemplo, ResNet152 obtuvo mejores resultados que todos los demás modelos de esa familia. Concretamente, DaViT-base obtuvo los mejores resultados y es el modelo más grande con el que se experimentó, con 88 millones de parámetros. Por otra parte, se encontró que los modelos más grandes suelen tener tiempos de inferencia mayores, lo cual los hace menos atractivos para aplicaciones de tiempo real. Además, se experimentó principalmente con dos arquitecturas de redes neuronales, las convolucionales, como ResNet, DenseNet e Inception y las *vision transformer* como ViT y DaViT. No se encontró una diferencia sustancial en cuanto a los resultados obtenidos por ambas arquitecturas, sin embargo cuando se tiene en cuenta la cantidad de parámetros y el tiempo de inferencia las arquitecturas convolucionales parecen más atractivas.

A partir de la generación de un sistema que permite generar reportes a partir de una video-inspección se encontró que las redes generadas tienen la capacidad de aportar valor a los procesos de trabajo de los profesionales del área. La herramienta generada tiene la capacidad de automatizar parte del proceso de trabajo actual de la empresa DICA & Asociados ahorrando tiempo de profesionales calificados, que de otra manera estaría siendo utilizado en tareas tediosas. Esto reafirma la utilidad de los modelos generados.

Reflexionando sobre los resultados, se concluye que las redes de clasificación son un modelo apropiado para el tratado del problema de detección de defectos en saneamiento.

Finalmente, el equipo concluye que existe potencial de mejora en varias aristas relacionadas a las decisiones que fueron tomadas en el proyecto. En la siguiente sección se elabora sobre las mismas.

5.2. Trabajo futuro

En esta sección se elabora sobre los diferentes puntos de mejora que se consideran en el trabajo realizado.

5.2.1. Detección de objetos

Como fue mencionado en la sección 4.1, se optó por tomar el camino de la clasificación de imágenes para afrontar el problema dado, esta decisión fue tomada debido a que no se disponía un conjunto de datos de tamaño y calidad

suficiente para detección de objetos. Si se dispone de dichos recursos, este es un enfoque que podría aportar modelos de interés para la investigación del grupo MINA.

Queda como posible trabajo futuro el probar esta vía, de forma tal que se pueda además de detectar la existencia de defectos, la ubicación de los mismos dentro de una imagen. Esto también podría mejorar la experiencia de la empresa DICA & Asociados, mejorando también la calidad de los reportes.

Es importante recalcar que para llevar a cabo este enfoque se debe disponer de un conjunto de datos de este tipo o crearlo. La segunda es una tarea que requiere muchos recursos, principalmente horas de trabajo de personal calificado en inspecciones de saneamiento.

5.2.2. Elección de modelos

Debido a la limitación de tiempo del proyecto, y a limitaciones del hardware disponible el equipo se limito a trabajar con modelos con menos de 88 millones de parámetros. Como se comentó en secciones anteriores, el entrenamiento de redes neuronales es un proceso que requiere de hardware especializado y que consume tiempo. En particular, la infraestructura utilizada ClusterUY (Nesmachnow y Iturriaga, 2019) tiene la limitación de que cada trabajo tiene un tiempo máximo de ejecución de 72 horas. El equipo se enfocó en trabajar con modelos que logren terminar su entrenamiento en un solo trabajo para facilitar la operativa. Además modelos con muchos parámetros exigen utilizar varias GPUs para su entrenamiento debido a la demanda de memoria RAM que tienen.

Se deja planteado como trabajo futuro experimentar con modelos más grandes. Tanto los resultados obtenidos como la investigación sobre el estado del arte apuntaron a que modelos con más parámetros pueden permitir obtener mejores resultados en las métricas. En la fuente (PapersWithCode, s.f.) se encuentran los mejores modelos del estado del arte en clasificación de imágenes, los modelos que figuran más alto en este rango suelen tener una gran cantidad de parámetros.

5.2.3. Creación de conjuntos de datos

Como continuación de los dos anteriores puntos, en una posible siguiente iteración de este proyecto se podría considerar la creación de un nuevo conjunto de datos, tanto si se decide por continuar por la línea de la clasificación de imágenes (5.2.2) o por pivotar y cambiar a la detección de objetos (5.2.1).

Como se mencionó previamente, en el caso de optar por la creación de un nuevo conjunto de datos, se deberá contar con el apoyo de expertos en el área debido a las complicaciones que pueden surgir a la hora de no solo clasificar los defectos, que ya de por sí es una tarea complicada si no se es conocedor de la materia, sino también de una posible delimitación de los mismos. Para esto último es aún más importante el hecho de contar con apoyo profesional, debido a la dificultad que surge a la hora de afrontar la problemática de la delimitación

de defectos, ya que en este aspecto puede haber varios puntos de vista inclusive si se cuenta con expertos en el área, ya que se tratan diferencias milimétricas.

5.2.4. Mejora del producto

Finalmente, se aborda el último punto, el de la mejora del producto. En esta etapa se pueden encontrar a su vez varios caminos para posibles mejoras del mismo.

En que refiere a la parte de preprocesado (la parte donde entra la inteligencia artificial) del producto, se pueden probar distintos modelos para el sistema, tanto si se decide utilizar alguno de los modelos presentados en este trabajo o por el contrario, si se decide utilizar algún otro modelo sea de clasificación de imágenes o de detección de objetos. También se puede probar la utilización de distintas heurísticas o mismamente la incorporación de odometría u otros datos del robot para la generación de reportes. A este último punto se le puede agregar la idea de diferenciar dos imágenes diferentes usando el ángulo del robot como referencia, o de validar la distancia recorrida con los datos del robot.

En cuanto al producto visto como pieza de software, queda como trabajo futuro el cambiar el resultado de salida del sistema de un video a una interfaz, que si bien es necesario que también contenga ese mismo video, esta debería ser interactiva por el usuario final. Finalmente en cuanto a la generación de reporte, se podría generar un mapa a modo gráfico del camino recorrido por el robot, de la misma forma que se hace en el flujo de trabajo de la empresa DICA & Asociados.

Como último punto, se puede intentar agregar la opción de realizar la inferencia en la nube, a algun servicio como lo puede ser por ejemplo AWS o Azure y así poder disminuir el tiempo de cómputo.

Referencias

Abdullah-Al-Wadud, M., Kabir, M. H., Akber Dewan, M. A., y Chae, O. (2007). A dynamic histogram equalization for image contrast enhancement. *IEEE Transactions on Consumer Electronics*, *53*(2), 593-600. doi: 10.1109/TCE.2007.381734

Amanatullah. (2023). *What are convolutional neural networks?* Internet. Descargado de https://medium.com/@amanatulla1606/fine-tuning-the-model-what-why-and-how-e7fa52bc8ddf

Asociados, D. . (s.f.). *Proyectos.* Internet. Descargado de https://dica.com.uy/proyectos

Bashar, A.-S. (2023). *Object detection and classification in videos using deep learning.* Internet. Descargado de https://medium.com/@basharsalaam/object-detection-and-classification-in-videos-using-deep-learning-9c822d3df6e4

Bhatia, R. (2017). *Top 5 image classification research papers every data scientist should know.* Internet. Descargado de https://analyticsindiamag.com/top-5-image-classification-research-papers-every-data-scientist-should-know/

Brownlee, J. (2020). *How to choose loss functions when training deep learning neural networks.* Descargado de https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., y Zagoruyko, S. (2020). *End-to-end object detection with transformers.*

Chen, D., He, M., Fan, Q., Liao, J., Zhang, L., Hou, D., ... Hua, G. (2019). Gated context aggregation network for image dehazing and deraining. En *2019 ieee winter conference on applications of computer vision (wacv)* (p. 1375-1383). doi: 10.1109/WACV.2019.00151

Cnn inception network. (s.f.). Internet. Descargado de https://datahacker.rs/deep-learning-inception-network/

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., y Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. En *2009 ieee conference on computer vision and pattern recognition* (p. 248-255). doi: 10.1109/CVPR.2009.5206848

Detection, S. D. (2023). *Sewer defects dataset* [Open Source Dataset]. https://universe.roboflow.com/sewage-defect-detection

-s68df/sewer-defects-u8zwz. Roboflow. Descargado de https://universe.roboflow.com/sewage-defect-detection-s68df/sewer-defects-u8zwz (visited on 2024-07-17)

Ding, M., Xiao, B., Codella, N., Luo, P., Wang, J., y Yuan, L. (2022). Davit: Dual attention vision transformers. En *European conference on computer vision* (pp. 74–92).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... others (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*. Descargado de https://arxiv.org/abs/2010.11929

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., y Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, *88*, 303–338.

GeeksforGeeks. (2024). *Multiclass classification vs multi-label classification.* Internet. Descargado de https://www.geeksforgeeks.org/multiclass-classification-vs-multi-label-classification/

Gholamalinejad, H. (2020). *Pooling methods in deep neural networks, a review.* Internet. Descargado de https://www.researchgate.net/figure/Example-of-Max-Pooling-operation_fig2_344277235

Hanley, P., y Drinkwater, A. (2020). *Pipeline assessment and certification program.* Editorial.

Haurum, J. B., y Moeslund, T. B. (2021). Sewer-ml: A multi-label sewer defect classification dataset and benchmark. En *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 13456–13467).

He, K., Zhang, X., Ren, S., y Sun, J. (2016). Deep residual learning for image recognition. En *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Huang, G., Liu, Z., van der Maaten, L., y Weinberger, K. Q. (2018). *Densely connected convolutional networks.*

IBM. (s.f.-a). *What are convolutional neural networks?* Internet. Descargado de https://www.ibm.com/topics/convolutional-neural-networks

IBM. (s.f.-b). *What is a neural network?* Internet. Descargado de https://www.ibm.com/topics/neural-networks

Introduction to residual networks. (2020). Internet. Descargado de https://www.geeksforgeeks.org/introduction-to-residual-networks/

Li, Y., Wang, H., Dang, L. M., Song, H.-K., y Moon, H. (2022). Vision-based defect inspection and condition assessment for sewer pipes: a comprehensive survey. *Sensors*, *22*(7), 2722.

MathWorks. (s.f.). *Multilabel image classification using deep learning.* Internet. Descargado de https://www.mathworks.com/help/deeplearning/ug/multilabel-image-classification-using-deep-learning.html

Matplotlib. (s.f.). Internet. Descargado de https://matplotlib.org/

Nassco. (s.f.). Internet. Descargado de https://nassco.org/

Nesmachnow, S., y Iturriaga, S. (2019). Cluster-uy: Collaborative scientific high performance computing in uruguay. En M. Torres y J. Klapp (Eds.), *Supercomputing* (pp. 188–202). Cham: Springer International Publishing.

Openpyxl. (s.f.). Internet. Descargado de https://openpyxl.readthedocs.io/en/stable/

PapersWithCode. (s.f.). *Image classification on imagenet.* Internet. Descargado de https://paperswithcode.com/sota/image-classification-on-imagenet

Patel, K. (2019). *Convolutional neural networks — a beginner's guide.* Internet. Descargado de https://towardsdatascience.com/convolution-neural-networks-a-beginners-guide-implementing-a-mnist-hand-written-digit-8aa60330d022

pirzada, M. H. (2023). *Difference between binary, multiclass, and multi-label classification.* Internet. Descargado de https://www.linkedin.com/pulse/difference-between-binary-multiclass-multi-label-pirzada/

Proyecto de grado - redes neuronales para detección de defectos en saneamiento. (2024). Internet. Descargado de https://gitlab.fing.edu.uy/santiago.acquarone/proyecto-de-grado-redes-neuronales-saneamiento

pyenv. (s.f.). Internet. Descargado de https://github.com/pyenv/pyenv

Pytorch. (s.f.). Internet. Descargado de https://pytorch.org/

Redmon, J., Divvala, S., Girshick, R., y Farhadi, A. (2016). You only look once: Unified, real-time object detection. En (pp. 779–788).

Roboflow. (s.f.). Internet. Descargado de https://roboflow.com/

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... others (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*, 211–252.

Saha, S. (2018). *A comprehensive guide to convolutional neural networks — the eli5 way.* Internet. Descargado de https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Sewage. (2024). *Sewage dataset* [Open Source Dataset]. https://universe.roboflow.com/sewage/sewage. Roboflow. Descargado de https://universe.roboflow.com/sewage/sewage (visited on 2024-07-17)

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015). Going deeper with convolutions. En *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1–9). Descargado de https://arxiv.org/abs/1409.4842

Terven, J., y Cordova-Esparza, D. (2023). A comprehensive review of yolo: From yolov1 to yolov8 and beyond. *arXiv preprint arXiv:2304.00501*.

timm. (s.f.). Internet. Descargado de https://timm.fast.ai/

Tkinter. (s.f.). Internet. Descargado de https://docs.python.org/es/3/library/tkinter.html

Umer, A. (2022). *Deep learning essentials.* Internet. Descargado de https://medium.com/codex/deep-learning-essentials-9cce7a911326

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). *Attention is all you need.*

Xie, Q., Li, D., Xu, J., Yu, Z., y Wang, J. (2019). Automatic detection and classi-

fication of sewer defects via hierarchical deep learning. *IEEE Transactions on Automation Science and Engineering*, *16*(4), 1836–1847.

Yoo, A. B., Jette, M. A., y Grondona, M. (2003). Slurm: Simple linux utility for resource management. En D. Feitelson, L. Rudolph, y U. Schwiegelshohn (Eds.), *Job scheduling strategies for parallel processing* (pp. 44–60). Berlin, Heidelberg: Springer Berlin Heidelberg.

Anexo A

Funcionamiento del programa *trainer*

A continuación se presentan tanto el funcionamiento general del programa *trainer*, así como algunos conceptos necesarios para el entendimiento de dicho programa. El programa *trainer* se encarga del entrenado de los distintos modelos elegidos.

A cada iteración del bucle de entrenamiento se le denomina época. Cada época tiene primero una fase de entrenamiento, donde se hace una pasada por cada imagen del conjunto de entrenamiento con el objetivo de aprender a clasificar las imágenes. Por consiguiente se tiene una fase de validación, donde se recorre el conjunto de validación con el objetivo de evaluar el aprendizaje realizado en la fase de entrenamiento. Este proceso, se debe hacer en un conjunto distinto al utilizado en el entrenamiento, de ahí el uso de un conjunto de validación. De esta forma, se observa si el error disminuye en un conjunto de imágenes que el modelo desconoce, y así evitar el sobreajuste sobre el conjunto de entrenamiento.

El programa permite elegir el máximo de épocas que se desea entrenar. Una vez que se cumple ese numero de épocas, la ejecución finaliza y se guardan los pesos aprendidos.

En la fase de entrenamiento, el modelo toma las imágenes del subconjunto de datos de entrenamiento de a lotes hasta recorrer todo el conjunto. El tamaño de los lotes es un parámetro del programa.

Se realiza inferencia para cada lote, obteniendo las predicciones del modelo para las entradas. Se calcula el error entre las predicciones del modelo y el etiquetado de las imágenes, denominado pérdida o *loss*. Este error se calcula utilizando la función de pérdida *Binary Cross Entropy*, la cual ha probado obtener buenos resultados para problemas de clasificación multi-etiqueta como lo es el caso de estudio. Para elegir esta función se consultó la fuente (Brownlee, 2020).

Se calculan los gradientes hacia atrás sobre los pesos, y se utiliza el optimizador para ajustar los pesos en base a los gradientes observados. Este ajuste es

lo que modifica a la red para que aprenda, y lo último que se hace en la fase de entrenamiento.

Luego, en la fase de validación, se toman las imágenes del conjunto de validación y se realiza inferencia usando al modelo con los pesos aprendidos en la fase anterior. Se calcula la pérdida sobre las imágenes del conjunto de validación y se promedia el resultado obtenido para las imágenes.

El modelo solo guarda los pesos aprendidos si estos logran obtener un error en validación mejor que los obtenidos en épocas anteriores. De esta forma, al final del entrenamiento se tiene guardados los mejores pesos, los que obtuvieron el menor error en el conjunto de validación.

Además se implementa un mecanismo de *early stopping*, que permite terminar el entrenamiento antes del máximo de épocas asignado si existe una tendencia a no mejorar al seguir entrenando. Este mecanismo funciona utilizando el valor de pérdida calculado sobre el conjunto de validación, y finaliza, de manera temprana, la ejecución de entrenamiento del modelo si este valor no mejora tras un número determinado de épocas. A este numero, se le denomina paciencia y es un parámetro del programa que por defecto tiene el valor de tres épocas.

Luego de la fase de validación termina la época y comienza la ejecución de la siguiente.

Al terminar la ejecución del bucle de entrenamiento, se tienen guardados los pesos con los que se obtuvo mejor rendimiento en el conjunto de validación.

Por último, el programa *trainer* implementa la evaluación de métricas sobre los modelos, en el subconjunto de evaluación. Estas son las métricas que utilizó el equipo para comparar de forma cuantitativa y objetiva el rendimiento de los modelos. Las métricas fueron utilizadas extensivamente en la sección de experimentación, y son descritas en la sección 4.3.2. El archivo de ejecución *trainer.py* se encuentra en el directorio *trainer* en el repositorio de este proyecto (*Proyecto de Grado - Redes neuronales para detección de defectos en saneamiento*, 2024).

Anexo B

Tiempos de inferencia para los modelos seleccionados

Se realizaron experimentos midiendo los tiempos de inferencia de los modelos seleccionados. Por un lado, se experimentó realizando la inferencia en CPU, buscando simular las condiciones en las que podría realizar la inferencia la empresa DICA & Asociados. Por otro lado, se experimentó realizando la inferencia en GPU, reflejando los resultados obtenibles cuando se dispone de hardware especializado.

Para ambos experimentos, se procesó inferencia en todos los cuadros de un video de inspección de la empresa DICA & Asociados de 10 segundos de duración y de 25 cuadros por segundo (FPS), y luego se calculó el tiempo de inferencia que demoró en promedio la inferencia de un cuadro. Para agregar redundancia a los experimentos, se realizó este cálculo 10 veces para cada modelo, y se tomó el tiempo promedio de todas las ejecuciones.

En cuanto al hardware utilizado, para el experimento haciendo inferencia en CPU se utilizo un sistema con un CPU Intel i5-11400 y 32 GB de memoria DDR4. Para el experimento haciendo inferencia en GPU se utilizó la infraestructura de ClusterUY (Nesmachnow y Iturriaga, 2019), en particular se realizó la inferencia en una GPU NVIDIA Tesla P100 con 16 GB de memoria HBM2.

En la tabla B.1 se presentan los resultados de estos experimentos.

Modelo	**Parámetros**	**Tiempo CPU**	**Tiempo GPU**
Inception v3	24M	0.0594	0.0267
ResNet50	26M	0.0481	0.0383
ResNet101	45M	0.0756	0.0231
ResNet152	60M	0.1000	0.0306
DenseNet121	8M	0.0477	0.0297
DenseNet161	29M	0.0939	0.0374
DenseNet201	20M	0.0748	0.0454
ViT-base	86M	0.0993	0.0341
DaViT-base	88M	0.1339	0.0413

Tabla B.1: Modelos seleccionados y los tiempos de inferencia en segundos para un cuadro.

Anexo C

Métricas utilizadas

Antes de describir las métricas utilizadas, es necesario primero explicar ciertos términos:

- Verdadero positivo: Clasificación en la que exitosamente se evalúa un caso positivo como tal. Se utiliza la abreviatura VP (o en inglés TP, de *True Positive*).
- Verdadero negativo: Clasificación en la que exitosamente se evalúa un caso negativo como tal. Se utiliza la abreviatura VN (o en inglés TN, de *True Negative*).
- Falso positivo: Clasificación en la que erróneamente se evalúa un caso negativo como uno positivo. Se utiliza la abreviatura FP (comparte sigla con la versión en inglés, *False Positive*).
- Falso negativo: Clasificación en la que erróneamente se evalúa un caso positivo como uno negativo. Se utiliza la abreviatura FN (comparte sigla con la versión en inglés, *False Negative*).
- Clasificación binaria: Clasificación que puede devolver dos valores: positivo o negativo.

Finalmente, en el contexto de las clasificaciones binarias tenemos las siguientes definiciones:

- Exactitud: Del inglés *accuracy*, es la medida que mide el porcentaje de aciertos de un modelo. Se calcula de la manera $\frac{VP+VN}{VP+VN+FP+FN}$.
- Precisión: Del inglés *precision*, es la medida que se utiliza para calcular la proporción de verdaderos positivos dentro de todos los clasificados como positivos. Se calcula como $\frac{VP}{VP+FP}$.
- Exhaustividad: Del inglés *recall*, es la medida que se utiliza para calcular la proporción de positivos que se evaluaron como tales. Se calcula como $\frac{VP}{VP+FN}$.

- Valor-F: Del inglés *F-score*, es una medida utilizada para combinar en cierta forma la precisión y la exhaustividad. Se calcula de la manera $(1 + \beta^2) * \frac{precisión * exhaustividad}{(\beta^2 * precisión) + exhaustividad}$, siendo β el valor F elegido, por ejemplo si se quiere tomar la métrica Valor-2 se debe sustituir F por 2.

  Nota: En todas las métricas nombradas se toma el promedio de forma no ponderada, es decir, se hace el promedio simple de los valores. Para la métrica de valor-F, se decidió calcular también el promedio ponderado según la cantidad de instancias de cada defecto, por lo que para el valor-F se va a tener la versión macro (no ponderada) y la versión micro (ponderada).

Anexo D

Resultados experimentales

A continuación se exponen los resultados de la etapa de experimentación que se presentó en el capítulo 4.3. Como se comentó en dicho capítulo, la experimentación contó de tres etapas, cada una de las cuales contó con un conjunto de imágenes de distinto tamaño. Aquí se muestran las métricas obtenidas para los distintos modelos que entrenados.

Nota: En la columna épocas se puede apreciar que varias filas tienen un asterisco (*) al lado de la época, lo esto que describe es que hubo *early stopping* (descrito en 4.2.3), lo que significa que hubo un detenimiento temprano al haber pasado tres épocas sin mejoramiento del valor de pérdida en el conjunto de validación.

Modelo	**Parámetros**	**Lote**	**Épocas**	**Exactitud**	**Precisión**	**Exhaustividad**	**Macro F1**	**Micro F1**
AlexNet	62M	64	21*	0.9531	0.7139	0.2995	0.3703	0.6265
ResNet-18	11M	64	10*	0.9570	0.6808	0.4363	0.5073	0.6779
ResNet-50	26M	64	8*	0.9583	0.6440	0.4949	0.5486	0.7037
ResNet-101	45M	64	8*	0.9585	0.6305	0.5003	0.5532	0.6931
InceptionV3	24M	32	13*	0.9569	0.6815	0.4043	0.4699	0.6703
DenseNet-121	8M	32	12*	0.9575	0.7016	0.3935	0.4789	0.6655
DenseNet-161	29M	32	10*	0.9572	0.6632	0.4308	0.4960	0.6822
DenseNet-201	20M	16	11*	0.9567	0.6722	0.3874	0.4772	0.6578
ViT-base	86M	64	3	0.9570	0.7695	0.3646	0.4457	0.6720
DaViT-base	88M	128	8*	0.9592	0.6856	0.4654	0.5393	0.6836

Tabla D.1: Resultados de los entrenamientos con 120.000 imágenes.

Modelo	Parámetros	Lote	Épocas	Exactitud	Precisión	Exhaustividad	Macro F1	Micro F1
ResNet-50	26M	64	10*	0.9639	0.7081	0.5399	0.6029	0.7337
ResNet-101	45M	64	9*	0.9649	0.7331	0.5225	0.5997	0.7367
InceptionV3	24M	32	21*	0.9641	0.7333	0.4926	0.5794	0.7257
DenseNet-121	8M	32	15	0.9629	0.7501	0.4929	0.5273	0.7147
DenseNet-161	29M	32	15	0.9642	0.7641	0.4817	0.5728	0.7220
DenseNet-201	20M	32	15	0.9638	0.7202	0.5097	0.5851	0.7245
ViT-base	86M	64	11*	0.9645	0.7730	0.4959	0.5900	0.7274
DaViT-base	88M	64	8*	0.9653	0.8113	0.4931	0.6013	0.7156

Tabla D.2: Resultados de los entrenamientos con 500.000 imágenes.

Modelo	Parámetros	Lote	Épocas	Exactitud	Precisión	Exhaustividad	Macro F1	Micro F1
ResNet-50	26M	64	12*	0.9685	0.7575	0.5690	0.6435	0.7628
ResNet-101	45M	64	12*	0.9693	0.7789	0.5815	0.6587	0.7692
ResNet-152	60M	64	12*	0.9690	0.7627	0.6025	0.6677	0.7707
InceptionV3	24M	32	15	0.9658	0.7515	0.5431	0.6121	0.7470
DenseNet-161	29M	32	15	0.9674	0.7868	0.5390	0.6264	0.7507
ViT-base	86M	64	8*	0.9626	0.8030	0.4293	0.5433	0.6876
DaViT-base	88M	128	9*	0.9702	0.7919	0.6050	0.6794	0.7790

Tabla D.3: Resultados de los entrenamientos con 1.170.000 imágenes.

Anexo E

Tiempos de ejecución del sistema

Se efectuaron mediciones de los tiempos de ejecución del sistema, precisamente, de los procesos de generación de documento y de video. Para ello, se considera un video-inspección de entrada de 11 minutos y 46 segundos de duración, brindado por DICA & Asociados, a partir del cual se generará un documento que reporte los defectos hallados y un video que exhiba gráficamente el proceso de clasificación del modelo. En este sentido, se seleccionó el modelo ResNet152 (uno de los que obtuvieron mejores resultados) para realizar el proceso de inferencia, obteniendo las predicciones utilizadas para la generación de documento y video. A fin de contemplar distintos escenarios, se ejecutaron ambos procesos en la infraestructura de ClusterUY (Nesmachnow y Iturriaga, 2019), utilizando una CPU Intel(R) Xeon(R) Gold 6138, y una GPU NVIDIA Tesla P100 con 16 GB de memoria HBM2.

Para cada proceso, se desglosa el tiempo que conlleva la inferencia del modelo (módulo de clasificación) y la construcción del documento (módulo de documento) o del video (módulo de video), según corresponda. El tiempo de inferencia es siempre el mismo, dado que ambos procesos precisan de las predicciones del módulo de clasificación.

Modelo	**Hardware**	**Módulo de clasificación**	**Módulo de documento**	**Total**
ResNet152	GPU	06:36	01:12	07:48
	CPU	01:06:23	01:06	01:07:29

Tabla E.1: Tiempos de ejecución para la generación de documento, utilizando el modelo ResNet152 para la inferencia del video de inspección de entrada.

Modelo	Hardware	Módulo de clasificación	Módulo de video	Total
ResNet152	GPU	06:36	03:58:51	04:05:27
	CPU	01:06:23	05:06:55	06:13:18

Tabla E.2: Tiempos de ejecución para la generación de video, utilizando el modelo ResNet152 para la inferencia del video de inspección de entrada.

Anexo F

Pseudocódigo función heurística

Se describe un pseudocódigo del funcionamiento de la función heurística. La función principal que se invoca es *obtener_cuadros_defectos* que a su vez invoca a las otras dos, *aplicar_heuristica* y *unificar_cuadros_defectos*.

```
DEFECTOS = ['defecto_1', 'defecto_2', ..., 'defecto_n']
FPS = m


FUNCION unificar_cuadros_defectos(cuadros, segundos_separacion):
    distancia = segundos_separacion * FPS
    subgrupos_cuadros = []

    subgrupo = [cuadros[0]]
    nro_cuadro = cuadros[0][0]
    PARA i EN rango(1, longitud(cuadros)):
        SI cuadros[i][0] - nro_cuadro <= distancia:
            subgrupo.agregar(cuadros[i])
        SINO:
            subgrupos_cuadros.agregar(subgrupo)
            subgrupo = [cuadros[i]]
            nro_cuadro = cuadros[i][0]

    subgrupos_cuadros.agregar(subgrupo)

    RETORNAR subgrupos_cuadros


FUNCION aplicar_heuristica(predicciones, umbral,
```

```
    segundos_intervalo, segundos_separacion):
    long_intervalo = segundos_intervalo * FPS
    cuadros_defecto = {}

    PARA i EN rango(longitud(predicciones)):
        tope = i + long_intervalo
        SI tope > longitud(predicciones):
            CORTAR
        SINO:
            promedio = suma(predicciones[i:tope]) / long_intervalo
            SI promedio > umbral:
                max_prediccion = max(predicciones[i:tope])
                nro_cuadro = indice(predicciones[i:tope], max_prediccion)
                cuadros_defecto[nro_cuadro] = max_prediccion

    cuadros_defecto = [(cuadro, cuadros_defecto[cuadro])
        PARA cuadro EN cuadros_defecto]
    grupos_cuadros = unificar_cuadros_defectos(
        cuadros_defecto, segundos_separacion)

    cuadros_unificados = []
    PARA grupo EN grupos_cuadros:
        nro_cuadro, prediccion = obtener_cuadro_max_prediccion(grupo)
        cuadros_unificados.agregar((nro_cuadro, prediccion))

    RETORNAR cuadros_unificados


FUNCION obtener_cuadros_defectos(predicciones, umbrales,
    segundos_intervalo, segundos_separacion):
    cuadros_a_reportar = []
    PARA i EN rango(longitud(DEFECTOS)):
        cuadros_defecto = aplicar_heuristica(predicciones[i],
            umbrales[i] segundos_intervalo, segundos_separacion)
        PARA cuadro_pred EN cuadros_defecto:
            cuadros_a_reportar.agregar((cuadro_pred, DEFECTOS[i]))

    cuadros_a_reportar = [(cuadro, cuadros_a_reportar[cuadro])
        PARA cuadro EN ordenado(cuadros_a_reportar)]
    grupos_cuadros = unificar_cuadros_defectos()

    cuadros_a_reportar = {}
    PARA grupo EN grupos_cuadros:
        primero = grupo[0][0]
        cuadros_a_reportar[primero] = [grupo[0][1]]
        PARA i EN rango(1, longitud(grupo)):
```

```
            cuadros_a_reportar[primero].agregar(
                grupo[i][1])

    RETORNAR cuadros_a_reportar
```

Anexo G

Comparativa visual de los reportes

A continuación se compara lado a lado los reportes provistos por la empresa DICA & Asociados con los generados por el sistema. Esta es una forma directa y visual de comparar los informes que permite observar si los defectos reportados por el sistema corresponden con los reportados por profesionales.

En la figura 4.17 se visualiza un ejemplo de un reporte generado por DICA & Asociados (izquierda) y por el sistema (derecha).

En la tabla G.1 se realiza una comparativa entre los defectos e imágenes obtenidos en los reportes de la figura 4.17, donde se compara imagen a imagen describiendo el tipo de defecto, en qué reporte se encuentra y el momento en el video de inspección.

	Tipo de defecto	**Reporte DICA & Asociados**	**Reporte sistema**	**Tiempo**
1	Grieta/Rotura	Figura 7	-	22:09:01
2	Grieta/Rotura	-	Figura 7	22:08:43
3	Grieta/Rotura	Figura 8	-	22:09:50
4	Junta desplazada	-	Figura 8	22:09:35
5	Grieta/Rotura	Figura 9	Figura 9	22:10:08
6	Junta desplazada	Figura 10	Figura 11	22:10:59
7	Grieta/Rotura	-	Figura 10	22:10:17
8	Junta desplazada	Figura 11	Figura 12	22:11:08
9	Grieta/Rotura	Figura 12	-	22:10:48

Tabla G.1: Tabla comparativa entre imágenes obtenidas de los informes generados por DICA & Asociados y por el sistema.

Observando los resultados de la tabla, se aprecia que el informe generado por el sistema reporta imágenes en la misma forma que el brindado por la empresa. Particularmente, tres de las seis imágenes reportadas por el sistema

corresponden a las mismas reportadas por la empresa, es decir, al mismo tipo de defecto e instante en el tiempo del video-inspección. Esto se ve reflejado en las filas cinco, seis y ocho de la tabla G.1, donde se tiene al mismo defecto en ambos reportes (no necesariamente en la misma posición).

Los reportes generados tanto por la empresa como por el sistema, poseen más de una página con imágenes de defectos. En la figura 4.17 y en la tabla G.1 se visualiza solo una página de estos reportes. Puede suceder entonces, que un defecto presente en la primera página del reporte generado por la empresa tenga una correspondencia con un defecto presente en la segunda página del reporte generado por el sistema. Por ejemplo, la *Figura 7* del reporte del sistema no tiene una correspondencia directa con las imágenes del reporte de la empresa que se visualizan en la figura 4.17, pero sí la tiene con la página anterior del reporte de la empresa.

Por defecto, el sistema tiende a detectar una mayor cantidad de defectos en comparación con los reportes generados por profesionales. De esta forma, se le da al operador la opción de eliminar aquellas que no desee, o eventualmente, modificar los parámetros de generación de reportes del sistema.