UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA
MAESTRÍA Y DOCTORADO EN CIENCIAS E INGENIERÍA

T Í T U L O:

SISTEMA DE DETECCIÓN Y CLASIFICACIÓN DE DEFECTOS SUPERFICIALES EN MATERIALES METÁLICOS PARA APLICACIONES DE INGENIERÍA MEDIANTE VISIÓN POR COMPUTADORA

TESIS PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS

PRESENTA:

PABLO JESÚS URIARTE DE LA CRUZ

DIRECTORA:

DRA. LIDIA ESTHER VARGAS OSUNA

CODIRECTOR:

DR. JOSÉ MANUEL RAMÍREZ ZARATE

MEXICALI, B.C. **NOVIEMBRE DE 2022**

AGRADECIMIENTOS

A mi directora de tesis, Dra. Lidia Esther Vargas Osuna, por haber dirigido este trabajo de investigación y por su invaluable apoyo y seguimiento durante el desarrollo del mismo.

A mis sinodales, Dr. José Manuel Ramírez Zarate, Dra. Virginia García Ángel, Dra. María Amparo Oliveros Ruíz y MAID. Jorge Miramón Angulo, por sus valiosas observaciones que ayudaron a mejorar este trabajo.

A la Universidad Autónoma de Baja California, por darme la oportunidad de realizar mis estudios de posgrado.

A las empresas Esliter de México, Precisión Sheet Metal y Materiales y Molduras de Mexicali, por abrirme sus puertas y permitirme hacer uso de sus instalaciones para la realización de pruebas, que fueron fundamentales para lograr los resultados obtenidos.

A mi familia, por siempre estar ahí, por el apoyo incondicional, la motivación de seguir adelante y siempre creer en mí.

A Dios, por darme salud y permitirme concluir con éxito mis estudios.

Pablo J. Uriarte

DEDICATORIA

A Dios, por prestarme vida y salud, lo que me ha permitido cumplir las metas que me he propuesto para superarme, agradecerle por todas las bendiciones que me ha brindado a lo largo de mi vida, especialmente por mi familia, que ha estado siempre conmigo apoyándome y dándome fortaleza de seguir adelante.

A mis padres, por creer en mí, por la formación que me han dado, por su esmero para siempre guiarme por el camino del bien, por educarme para ser una mejor persona y por siempre contar con su apoyo y comprensión en todas mis decisiones.

A mi esposa, por su apoyo incondicional, su motivación, sus consejos, la confianza depositada en mí y sobre todo por su paciencia durante toda mi formación, lo que me ha dado la fortaleza suficiente para seguirme superando y lograr concluir con éxito todas mis metas.

A mis seres queridos, que se han vuelto parte de mi familia y me han abierto las puertas de su hogar, con los que he compartido alegrías y tristezas, y que han estado siempre ahí para ofrecerme su cariño, su apoyo, sus mejores deseos y motivación para seguir adelante y ser mejor cada día.

A todos ustedes, con cariño.

Pablo J. Uriarte

RESUMEN

El presente trabajo tiene como objetivo, el diseño y desarrollo de un sistema para la identificación y clasificación en tiempo real de defectos superficiales presentes en materiales metálicos, utilizando el procesamiento de imágenes digitales mediante técnicas de transferencia de aprendizaje (Transfer Learning).

La detección automática y el reconocimiento de defectos superficiales en la industria son objetivos para el desarrollo de nuevas tecnologías en el ámbito de la visión por computadora, con la finalidad de obtener un mayor control de calidad y ventajas competitivas en la producción, por lo tanto, el sistema desarrollado realiza la detección oportuna de defectos superficiales presentes durante el proceso de manufactura de estos materiales.

Utilizando un sistema de captura de imágenes, se llevó a cabo el procesamiento de las mismas utilizando los modelos de redes neuronales existentes, previamente entrenados, con el fin de que el sistema fuese capaz de reconocer ciertos patrones en las imágenes, detectar y clasificar todo aquel defecto superficial para el cual fue entrenado.

ABSTRACT

The target of this work is the design and development of a system for real time identification and classification of surface defects present on metallic materials, using digital image processing through transfer learning techniques.

Automatic detection and recognition of surface defects in the industry are objectives for the development of new technologies in computer vision field in order to obtain greater quality control and competitive advantages in production, therefore, the developed system performs timely detection of surface defects present during manufacturing process of these materials.

Using an image capture system, images were processed using existing previously trained neural network models, so that the system was capable of recognizing certain patterns in the images, detecting and classifying all those surface defect for which he was trained.

ÍNDICE DEL CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ABREVIACIONES

END	Ensayos No Destructivos
AMEXEND	Asociación Mexicana de Ensayos No Destructivos
ASNT	Sociedad Estadounidense de Pruebas No Destructivas (*American Society for Nondestructive Testing*)
RNC	Redes Neuronales Convolucionales
IA	Inteligencia Artificial
MCP	Modelo McCulloh y Pitts
GPU	Unidad de Procesamiento Gráfico (*Graphics Processing Unit*)
CNN	Red Neuronal Convolucional (*Convolutional Neural Network*)
COCO	Objetos Comunes en Contexto (*Common Objects in Context*)
ILSVRC	Desafío de reconocimiento visual a gran escala de ImageNet (*ImageNet Large Scale Visual Recognition Challenge*)
SVM	Soporte de Máquinas de vectores (*Support Vector Machine*)
ROI	Regiones de Interés (*Regions of Interest*)
R-CNN	Redes Neuronales Convolucionales Basadas en Regiones (*Region Based Convolutional Neural Networks*)
RPN	Red de Propuesta de Región (*Region Proposal Network*)
R-FCN	Redes Totalmente Convolucionales Basadas en Regiones (Region-Based Fully Convolutional *Networks*)
SSD	Detector de Disparo Único (*Single Shot Detector*)
VGG16	Grupo de Geometría Visual (*Visual Geometry Group*)
YOLO	Solo Miras Una Vez (*You Only Look Once*)
DPM	Modelos de Piezas Deformables (*Deformable Parts Models*)
AP	Precisión Promedio (*Average Precision*)
mAP	Precisión Media Esperada (*Mean Average Precision*)
IOU	Intersección Sobre Unión (*Intersection Over Union*)
TP	Verdadero Positivo (*True Positive*)
FP	Falso Positivo (*False Positive*)

FN	Falso Negativo (*False Negative*)
GPS	Sistema de Posicionamiento Global (*Global Positioning System*)
URL	Localizador Uniforme de Recursos (*Uniform Resource Locator*)
HTML	Lenguaje de marcado de hipertexto (HyperText Markup Language)
CSS	Hojas de estilo en cascada (*Cascading Style Sheets*)
SDK	Kit de desarrollo de software (*Software Development Kit*)
API	Interfaz de programación de aplicaciones (*Application Programming Interface*)
NDK	Kit de desarrollo nativo (*Native Development Kit*)
GUI	Interfaz gráfica del usuario (*Graphical User Interface*)
XMPP	Protocolo extensible de mensajería y presencia (*Extensible Messaging and Presence Protocol*)
OpenGL	Biblioteca de gráficos abierta (*Open Graphics Library*)
SSL	Capa de sockets seguros (*Secure Sockets Layer*)
HAL	Capa de abstracción de hardware (*Hardware Abstraction Layer*)
OTA	Por el aire (*Over the Air*)
BSD	Distribución de software de Berkeley (*Berkeley Software Distribution*)
MPEG	Grupo de expertos en imágenes en movimiento (*Moving Picture Experts Group*)
MVC	Controlador de vista de modelo (*Model View Controller*)
SQL	lenguaje de consulta estructurado (*Structured Query Language*)
IDE	Entorno de desarrollo integrado (*Integrated Development Environment*)
XAML	Lenguaje de marcado de aplicaciones extensible (*Extensible Application Markup Language*)
XNA	Arquitectura de Sección Transversal de Próxima Generación (*Cross Section Next Generation Architecture*)
ANT	Otra herramienta ordenada (*Another Neat Tool*)
SVN	Control de subversión (*Sub Version Control*)
JVM	máquina virtual de Java (*Java Virtual Machine*)
VOC	Clases de objetos visuales (*Visual Object Classes*)
CUDA	Arquitectura unificada de dispositivos (*Unified Device Architecture*)

GPU	Unidad de procesamiento gráfico (*Graphics Processing Unit*)
TPU	Unidad de procesamiento de tensores (*Tensor Processing Unit*)
CuDNN	Biblioteca de redes neuronales profundas (*Deep Neural Network Library*)
CSV	Valores Separados por Comas (*Comma Separated Values*)
RAM	Memoria de acceso aleatorio (*Random Access Memory*)
NMS	Supresión no máxima (*Non Max Suppression*)

CAPÍTULO 1
INTRODUCCIÓN

1.1. INTRODUCCIÓN

En la última década, los sistemas de visión artificial se han convertido en un elemento clave para el control de calidad de diversos sistemas productivos [1]. En este campo, las dos aplicaciones más importantes de la visión, en un sentido amplio, son la inspección visual automática y el montaje automático (robótica).

La inspección visual automatizada ha tenido muchas y variadas aplicaciones. En el sector industrial, con la utilización de materiales metálicos como el acero y el aluminio, se fabrica una gran cantidad de productos utilizados en diversas aplicaciones ingenieriles, que necesitan ser analizadas de manera integral por inspectores humanos [2]. La inspección de estos productos es una tarea muy tediosa y repetida que se puede enfatizar para aquellos que realizan esta labor. El resultado final de este tipo de inspección humana puede variar, dependiendo de las emociones, las condiciones ambientales y otros factores subjetivos casi inevitables. Además, en ocasiones, debido a la velocidad del proceso o a la necesidad de analizar aspectos no diferenciales con el ojo humano [3], ni siquiera un operador puede realizar una inspección visual directa.

En este sentido, los sistemas de visión artificial parecen ser una posibilidad muy atractiva para mejorar este proceso. La introducción de esta tecnología permite realizar pruebas objetivas de todos los productos, lo que mejora la calidad y reduce los costos de producción. En los últimos años, los sistemas de inspección visual automatizados han demostrado sus ventajas en las líneas de producción de una amplia variedad de productos, gracias al desarrollo de potentes herramientas a precios competitivos [4]. Sin embargo, estas aplicaciones no han llegado por completo a la industria del acero y el aluminio, ya que la inspección de superficies metálicas suele ser difícil debido a la variedad de superficies reflectantes y defectos que se encuentran en ellas.

1.2. PROBLEMÁTICA

En cualquier proceso de fabricación, es posible que se presenten defectos en el producto debido a un gran número de situaciones que se presentan durante su elaboración, y para resolver tales situaciones, los ingenieros optan por considerar la inspección de los productos durante su proceso de fabricación.

La inspección visual puede dar al fabricante cierta certidumbre del producto fabricado, porque se asegura que alguien ha observado el producto antes de empaquetarlo y enviarlo desde la planta de fabricación hacia su destino. Algunos fabricantes incluso incluyen el nombre de la persona que probó el producto en un sello y/o inscripción en sus productos.

Se puede suponer que, al inspeccionar el producto, el inspector podrá detectar defectos que el fabricante ni siquiera había pensado o imaginado, pero la situación real es diferente, las pruebas las realizan personas y cada una tiene diferentes habilidades y conocimientos, por lo que cierto defecto es aceptable para uno, pero no para otro. Estas diferencias se denominan diferencias en los criterios de prueba.

La forma de compensar e intentar que todos los inspectores tomen aproximadamente las mismas decisiones es ayudarlos con capacitación y ayudas visuales, están capacitados para comparar lo que ven, con una foto de referencia que se considera un límite aceptable. Esto les proporciona una referencia de lo que deben identificar en los productos, como el número máximo de rayones permitidos o la gravedad de la deformación visible de la pieza.

El problema con este enfoque es que los criterios se actualizan constantemente a medida que se agregan nuevas características, lo que crea una cantidad infinita de aspectos para que el revisor busque y encuentre, y el tiempo disponible para la revisión es limitado debido a la presión constante para probar más piezas. Incluso el personal bien capacitado, no puede recordar con precisión todos los criterios al inspeccionar piezas, lo que permite que las piezas defectuosas escapen del proceso de inspección.

Otro factor que afecta la calidad del producto es el cambio de referencia, en cuyo caso, el inspector puede cambiar fácilmente el estándar para que sea más suave o más estricto sin darse cuenta. En general, los estándares se suavizan cuando los inspectores están bajo presión para terminar a tiempo, requieren aumentar la producción de piezas, están más cerca

del final del turno, o debido a la proximidad del horario de descanso, por nombrar algunos ejemplos. Por otra parte, las normas tienden a volverse más estrictas cuando la presión está en la calidad, por ejemplo, cuando se rechazan productos y se advierte a los inspectores para que tomen las acciones necesarias.

El problema para las empresas cuando se presentan estas variaciones en la calidad de sus productos impacta directamente al costo, se origina desperdicio de productos, retrabajo y desabasto innecesario del producto en un momento determinado. Por otro lado, se corre el riego con el cliente de proveerle productos con características inaceptables, lo que también genera costos por reposición del producto.

Hay algunos otros factores que también contribuyen a la diferencia en los estándares, como la llamada ceguera de taller, que se manifiesta cuando el cerebro humano deja de notar la diferencia después de revisar el mismo producto durante unas horas y entra en un bucle donde ya no logra identificar diferencias entre los productos inspeccionados, por lo que existe el riesgo de que el inspector permita pasar un artículo dañado.

El último factor a mencionar es la variación del producto, si la línea de producción tiene una pequeña cantidad de productos diferentes, no es difícil para un inspector capacitado recordar las diferencias entre la gama de productos que deben conocerse, pero si tiene una amplia variedad de productos y si la variación entre cada producto es pequeña, los inspectores pueden tener dificultades para distinguir entre piezas defectuosas y aceptables.

En esta investigación se analizó un área de manufactura donde existe un gran número de productos, y todos ellos están propensos a presentar defectos en su acabado final al llegar a la última etapa del proceso. Durante la fabricación de materiales para la elaboración de estructuras, es común contar con una amplia variedad de productos que varían en función a la forma (tubular redondo o cuadrado), su dimensión (½’, ¾’, 1’, 1-¼’, 1-½’, por mencionar algunas dimensiones), el calibre de la lámina (24, 22, 20, 18, entre otros calibres), por lo que debido al gran número de piezas que se fabrican y a las variantes que estos materiales pueden presentar, es muy difícil para el inspector a cargo llevar a cabo una inspección visual de la totalidad de las piezas.

En ocasiones, el proceso de manufactura de estos materiales continua sin la debida supervisión e inspección, lo que origina que piezas que pueden presentar algún defecto en su acabado o forma pasen a la siguiente etapa del proceso, donde son preparadas para su embalaje y distribución.

Estos productos se someten a varias inspecciones visuales durante distintos puntos del proceso, pero la más importante es al final, cuando se inspeccionan en busca de defectos en la superficie o acabado del material, por lo que es un desafío lograr una inspección 100 % detallada de todos los aspectos del acabado en la totalidad de las piezas que salen de la línea de producción.

De hecho, debido a la gran cantidad de productos y la rapidez del proceso, las inspecciones se realizan de forma aleatoria, los inspectores toman un conjunto de materiales, los organizan y empiezan a buscar diferencias entre ellos, examinan cada una de las caras del material hasta terminar la inspección.

Esta es una situación compleja ya que las principales quejas de los clientes son originadas por materiales que presentan defectos superficiales visibles, como perforaciones, abolladuras o discontinuidad en las uniones que se realizan mediante un proceso de soldadura por inducción, lo que compromete las propiedades estructurales del material.

1.3. JUSTIFICACIÓN

El control de calidad es estrictamente requerido en la industria para vender productos de calidad que cumplan con todos los aspectos técnicos impuestos asociados con ella: diseño, propiedades físicas, propiedades mecánicas, acabado superficial, etc. [5].

Para lograr esta calidad se realizan diversos tipos de pruebas o inspecciones para asegurar las propiedades deseadas del producto. El desarrollo industrial ha creado la necesidad de probar y verificar sistemas, estructuras, equipos, partes y ensambles utilizando diversos métodos y técnicas especializadas, incluidos los ensayos no destructivos (NDT), como la inspección visual de los productos.

La inspección de los materiales es clave para mejorar la calidad del producto y dado que los clientes se muestran reacios a seguir aceptando productos defectuosos, es necesario minimizar la incidencia de material devuelto por parte de los clientes, por lo que es especialmente importante implementar un sistema que ayude a mejorar las condiciones de inspección actuales, reduciendo el material defectuoso que llega a los clientes y que causa demoras y costos de reemplazo.

Dada la alta rotación de operadores y el comportamiento estacional de los volúmenes de producción, existe una falta de habilidades y experiencia de los operadores, combinado con un proceso altamente manual y dependiente del operador, motivo por el cual se percibe un incremento de material con defecto.

Uno de los principales beneficios esperados del uso de sistemas automatizados radica en que una vez que se programan los estándares, estos tendrán una variación mínima y una repetibilidad constante, dado que es un algoritmo el que define los límites aceptables, siendo más preciso distinguir entre partes buenas y malas.

Otra ventaja significativa es la cantidad de elementos a inspeccionar, mientras que un inspector solo considerará una cantidad limitada de características, en un sistema automatizado se pueden programar según sea necesario, la única limitación es el tiempo total de inspección, que agregará algunos milisegundos. Con una base de datos de diferentes productos, cambiar el modelo en el sistema es un proceso simple.

1.4. HIPÓTESIS

Con el desarrollo de un sistema de inspección automatizado, se busca lograr una identificación temprana de los defectos superficiales en materiales metálicos, que se originan durante el proceso de fabricación, los cuales suelen repetirse y afectar a todo un lote de producción, además de ayudar al equipo de producción a eliminar errores y defectos posteriores mediante acciones preventivas.

Con el uso efectivo del sistema se prevé una mejora en los resultados de calidad y una reducción en la cantidad de material defectuoso que sale de la línea de producción y es entregado al consumidor final, lo que origina una reducción de costos por rechazos, además de evitar comprometer las propiedades estructurales del material que son requeridas para su aplicación final. El uso de estos materiales es muy extenso en la industria, comprende la elaboración de estructuras para vehículos, aeronaves, edificaciones, construcciones, por mencionar algunas de sus aplicaciones.

1.5. OBJETIVO GENERAL

El objetivo de esta investigación es el desarrollo de un sistema de inspección automatizado con la capacidad de identificar y clasificar en tiempo real defectos en superficies metálicas, que sean perceptibles a simple vista, como perforaciones, abolladuras o discontinuidades, los cuales se presentan comúnmente durante los procesos de transformación de lámina de acero utilizada como materia prima en la fabricación de materiales estructurales para diferentes aplicaciones de la industria.

1.5.1. Objetivos específicos

- Seleccionar una plataforma de transferencia de aprendizaje adecuada para el desarrollo del proyecto.
- Implementar en la plataforma seleccionada una red pre entrenada dedicada al reconocimiento de objetos en imágenes.

- Crear un conjunto de datos de los productos sujetos a inspección, así como de las características a identificar en cada uno de estos.
- Desarrollar un sistema para la identificación de defectos que se ejecute desde un dispositivo móvil con una efectividad superior al 70%.
- Comprobar el funcionamiento en un entorno industrial, procesando imágenes en tiempo real, identificando oportunamente los defectos que surgen durante el proceso de fabricación.

1.6. METODOLOGÍA

Para lograr estos objetivos, esta investigación se dividió en 3 etapas, las cuales se describen a continuación:

1. Revisión bibliográfica

Durante esta etapa, se llevó a cabo el estudio de trabajos de investigación relacionados al desarrollo de sistemas de visión por computadora, con el fin de recopilar información útil para la formulación del diseño y el desarrollo del sistema.

2. Diseño y desarrollo del sistema

Esta etapa comprendió las actividades relacionadas al desarrollo del sistema, las cuales fueron:

- La adquisición de imágenes de la superficie de los materiales, que hizo posible la compilación de un set de datos para la identificación y clasificación de las características a identificar.
- La selección de un modelo de detección de objetos, para la identificación de los defectos superficiales en los materiales, considerando aspectos como: eficiencia del algoritmo, velocidad de procesamiento y capacidades técnicas de los equipos utilizados.
- El desarrollo de una aplicación móvil para la detección de objetos.

- Comprobación de la funcionalidad en un entorno industrial.

3. Análisis y resultados

Finalmente se realizaron pruebas de campo con los materiales a inspeccionar para evaluar su funcionamiento.

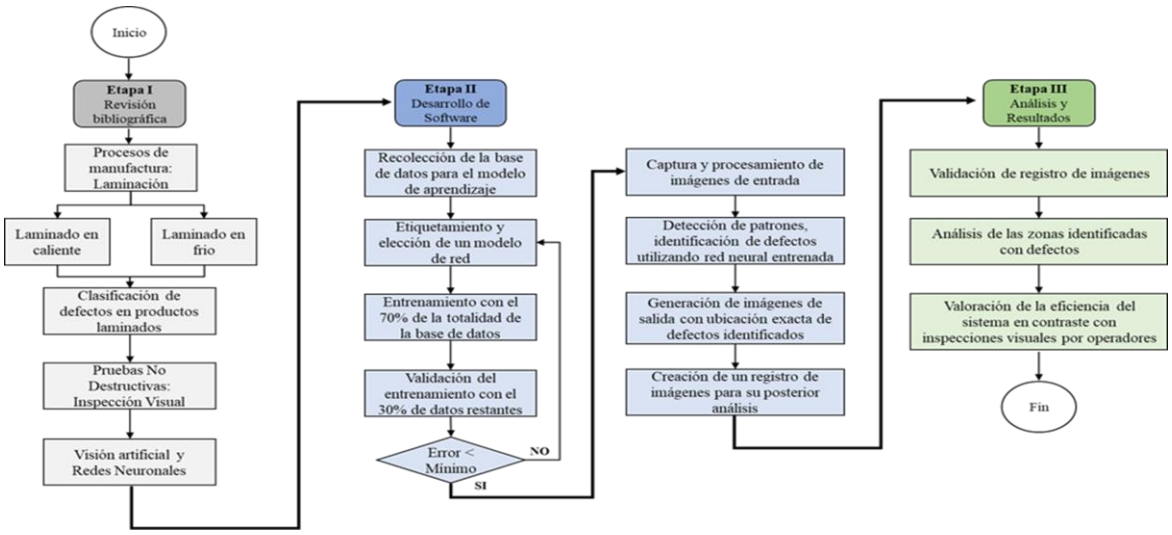

Figura 1. 1. Diagrama de flujo.

CAPÍTULO 2
ESTADO DEL ARTE

2.1. PROCESOS DE MANUFACTURA SIN ARRANQUE DE VIRUTA

Son un conjunto de acciones requeridas para cambiar las propiedades de la materia prima. Estas propiedades pueden ser diferentes, como la estética, tamaño, resistencia, forma o densidad y son realizadas dentro de la industria. Durante un proceso de fabricación se llevan a cabo una serie de actividades, como la extracción de la materia prima, su transformación y comercialización. Los procesos de fabricación que no eliminan las virutas son los que se realizan en la industria y no dejan residuo sólido. Los principales procesos productivos de esta clasificación se indican brevemente a continuación [6]:

Los procesos considerados como sin arranque de viruta son:

- Sinterizado
- Fundición
- Trefilado
- Doblado
- Embutición
- Forja
- Rolado
- Laminado

2.1.1. Sinterizado

La sinterización es el tratamiento de un polvo sólido precompactado a temperaturas por debajo de su punto de fusión de sus principales componentes, donde sus partículas se mantienen unidas por el fenómeno de soldadura. En la producción de cerámica, este proceso térmico convierte los productos en polvo en productos densos y uniformes. Se usa ampliamente para producir alúmina, berilio, ferrita y titanato en forma cerámica (Figura 2.1) [6].

Figura 2. 1. Balata para freno elaborada de material cerámico [6].

2.1.2. Fundición

Consiste en inyectar un material a un molde donde posteriormente se solidifica lo que permite elaborar formas complejas de tamaños diferentes. Los moldes para fundir deben hacerse de acuerdo a la forma y tolerancias de la pieza, porque la pieza se encoge al enfriarse. El material del molde debe ser refractario, el equipo debe tener la temperatura adecuada y se deben agregar los venteos adecuados para ventilar el aire y los gases. En la Figura 2.2 se puede apreciar un ejemplo de la fundición con arena. El molde debe permitir la extracción del colado para su posterior manipulación para eliminar los excesos [6].

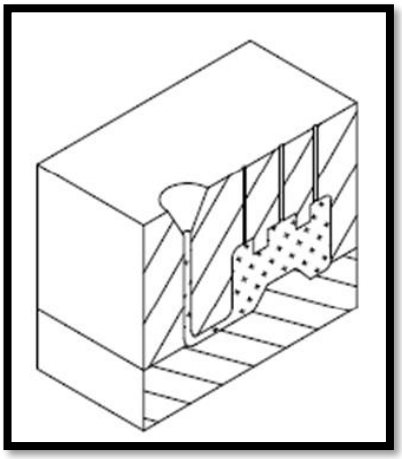

Figura 2. 2. Fundición [6].

2.1.3. Trefilado

Este es un proceso de conformado en frío que consiste en reducir la sección transversal de una varilla o alambre pasándolo a través de una perforación cónica hecha con un troquel. Los materiales que comúnmente son utilizados son: cobre, aluminio, acero y latón, o cualquier otro metal maleable. En la Figura 2.3 se ilustra un ejemplo de este movimiento. El propio trefilado, incluido el trefilado en frío, se pasa a través de un troquel de carburo de tungsteno, troquel o troquel de diámetro progresivamente menor en etapas sucesivas. Esta reducción de la sección transversal confiere al material una cierta dureza, lo que es beneficioso para sus propiedades mecánicas [6].

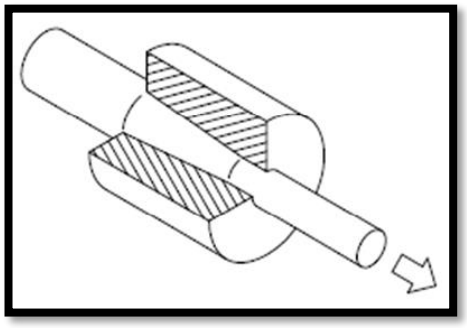

Figura 2. 3. Trefilado [6].

2.1.4. Doblado

Cuando se trabaja con metal laminado, la flexión se considera como la deformación de un metal en torno a un eje recto. Un ejemplo del proceso de flexión se puede apreciar en la Figura 2.4. Durante la operación de flexión, el metal se comprime en el plano neutro, mientras que el metal exterior se comprime bajo tensión. Los metales se deforman plásticamente, por lo que la curvatura adquiere una forma permanente cuando se elimina la tensión que la causó [6].

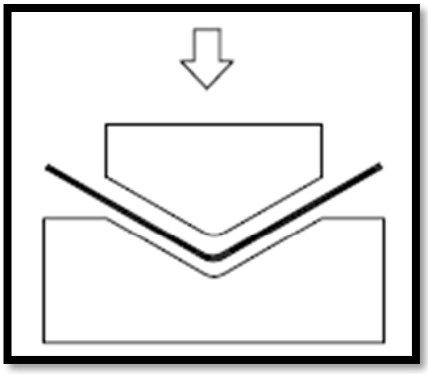

Figura 2. 4. Doblado [6].

2.1.5. Embutición

Este es un proceso mecánico en el que se forma una lámina delgada por presión en una forma hueca; esto se hace con una prensa donde se pega un sello con la forma interior de la pieza a copiar y luego se prensa la hoja hasta que tenga la forma como el punzón. En la Figura 2.5 se ilustra un caso de embutición profunda [6].

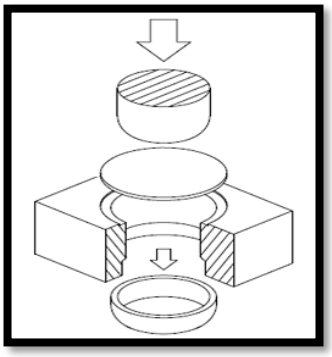

Figura 2. 5. Embutición [6].

2.1.6. Forja

La forja consiste en deformar un material que se comprime utilizando el impacto con otro material. El proceso facilita la producción de un gran número de piezas de alta resistencia para aplicaciones automotrices y aeroespaciales por mencionar algunas. En la Figura 2.6 se muestra cómo se lleva a cabo el proceso de forja. Un ejemplo gráfico de lo que sucede.

También podemos ver que la industria del acero y otros metales comunes utilizan la forja para fijar una forma básica a partir de piezas más grandes, que luego se mecanizan para obtener la forma y el tamaño final. La forja se lleva a cabo de diversas formas, la mayoría de las operaciones se realizan a alta temperatura, teniendo en cuenta la deformación requerida para el proceso y la necesidad de reducción de la resistencia y aumentar la flexibilidad del metal. La ventaja de la forja en frío es una mayor resistencia del componente debido al endurecimiento por deformación [6].

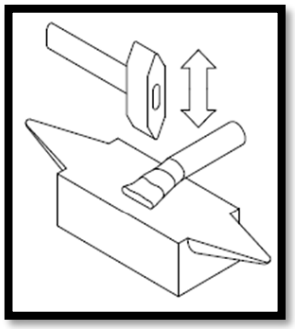

Figura 2. 6. Forjado [6].

2.1.7. Laminado

El laminado es un proceso en el cual el espesor inicial de la pieza es reducido por la fuerza de compresión ejercida sobre esta por dos rodillos (un ejemplo de laminado se ilustra en la Figura 2.7). Cada rodillo gira en direcciones opuestas, lo que hace que la pieza fluya entre ellos, aplicando fuerzas y creando fuerzas de corte por fricción entre cada rodillo y la pieza. La laminación requiere grandes inversiones de capital, por lo que los trenes de laminación se utilizan para producción en grandes volúmenes de producto (placas, láminas, por mencionar algunos.) [6].

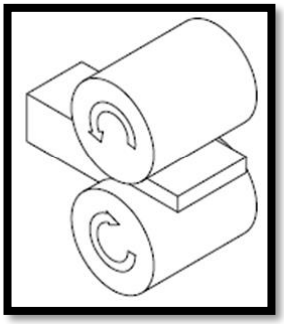

Figura 2. 7. Laminado [6].

2.1.8. Rolado

El rolado consiste en un proceso mecánico de plegado, básicamente se deforma la chapa para posteriormente darle forma de superficie plegada, lo que permite la formación de formas que se pueden dibujar sobre una superficie plana sin deformarse, Figura 2.8 [6].

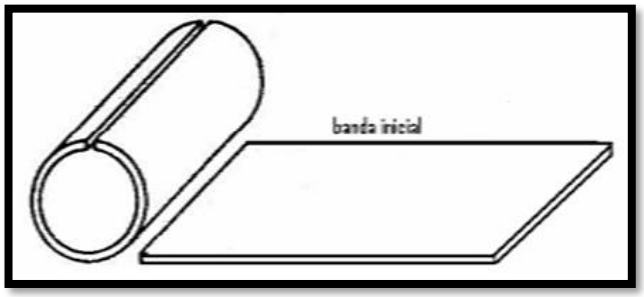

Figura 2. 8. Rolado [6].

2.2. PROCESO DE LAMINACIÓN

2.2.1. Definición de laminación

El laminado a un proceso común en la industria que consiste en reducir el espesor de láminas las cuales pueden ser de metal o materiales similares, mediante la aplicación de presión a través de diversos procesos como el laminado de perfiles o el de anillo. Por lo cual, se considera adecuado en materiales con buena flexibilidad. Las máquinas que realizan este proceso se denominan trenes de laminación. Los rodillos giran en direcciones opuestas, obligando al material a fluir entre ellos y aplicando fuerzas de compresión y cizallamiento debido a la fricción entre los rodillos y el metal [7].

La operación básica es laminación plana, y los productos laminados son planos. En general, se considera que estas placas tienen un grosor superior a 6 mm y se utilizan en estructuras tales como armazones de máquinas, cascos, calderas, puentes y barcos nucleares (Figura 2.9).

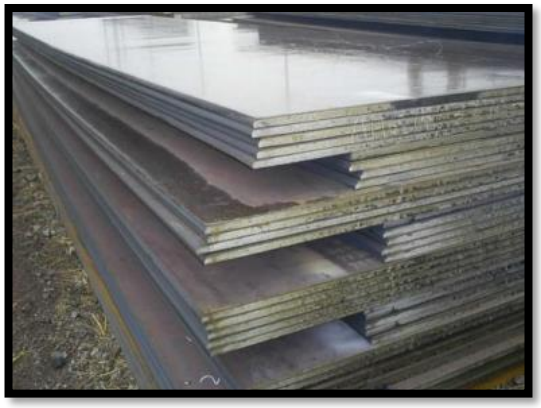

Figura 2. 9. Placas de acero obtenidas mediante proceso de laminación [7].

Las hojas suelen tener menos de 6 mm, están hechas de materias primas intermedias en piezas planas o rollos para su posterior procesamiento en varios productos. Se utilizan en carrocerías y fuselajes, electrodomésticos, envases de alimentos y bebidas, así como equipos de cocina y oficina (Figura 2.10).

Figura 2. 10. Rollo de hoja de acero obtenido mediante proceso de laminación [7].

2.2.2. Antecedentes históricos

Durante mucho tiempo, la idea fue hacer rodar metal pasándolo entre dos cilindros, cada uno girando en direcciones opuestas. Leonardo da Vinci tiene un boceto de 1497 de este tipo de máquina manual. En 1553, el francés Broaria usó una placa de impresión cilíndrica para obtener materiales laminados. Estas placas de capa crean láminas de metal gruesas

uniformes para hacer monedas de peso uniformes. Poco después, los herreros adoptaron los laminadores llamados "fanderías" (Figura 2.11) para reemplazar el trefilado o laminado manual accionado hidráulicamente.

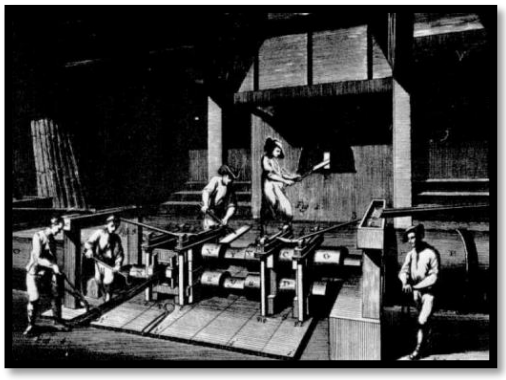

Figura 2. 11. Fandería (tren de laminación) [7].

Las llamadas "fanderías" fueron el primer paso en la especialización de los trenes de laminación, popularizándose a partir de 1700. Las técnicas de laminación fueron perfeccionadas por Polhelm en Suecia, Chopitels en Francia, Henry Court en Gran Bretaña, William Emerson y otros. En 1783, Henry Cort introdujo técnicas de laminación para formar acero. Su primer tren de laminación tenía rodillos acanalados que producían barras de refuerzo más rápidamente y a menor costo que el antiguo método de forjado. Inicialmente, las láminas de metal se introducen en moldes que les dan la forma deseada. Con el tiempo, estos troqueles son reemplazados por rodillos que deforman gradualmente el metal. La primera generación de máquinas laminadoras modernas estuvo en uso comercial en 1921 (Figura 2.12).

Figura 2. 12. Tren de laminación [7].

Marcos de puertas y ventanas, tubos de radiadores para automóviles, recortes de molduras y soportes de techo son algunos de los productos que comenzaron a fabricarse mediante perfilado en la década de 1940. Los equipos mejorados, los grandes avances en las materias primas y una mejor comprensión de la tecnología de laminación han ayudado a que el proceso se utilice para producir componentes para una variedad de aplicaciones. Actualmente, la mayor parte del metal producido pasa por el tren de laminación en al menos una etapa de producción [7].

2.2.3. Deformación en la laminación

La deformación que se produce durante el laminado se puede considerar bidimensional. Como resultado de la disminución del espesor, el material se alarga con un ligero aumento de la anchura. El término alargamiento suele referirse a la relación de la longitud de la barra antes y después de pasar por el espacio entre los rodillos. Debido al alargamiento del metal, el metal sale del rollo más rápido de lo que entra. Solo en el punto de contacto entre los rodillos y la pieza la velocidad periférica de la superficie del rodillo será igual a la velocidad direccional de la pieza, este punto se denomina punto neutro o punto de deslizamiento cero.

Los estudios muestran que la deformidad de la presión de la capa es causada por una combinación de compresión e inserto de corte. Por supuesto, el trabajo de compresión se debe al efecto de compresión adoptado por el cuerpo cilíndrico y el trabajo de corte por la fricción entre el cilindro y la pieza de trabajo.

2.2.4. Clasificación de los procesos de laminación de acuerdo a la temperatura

Estos procesos se dividen en laminación en frío y en caliente, teniendo en cuenta la temperatura a la que se encuentra la pieza de trabajo durante las operaciones de perfilado.

2.2.4.1. Laminado en caliente

Cuando la temperatura al que es sometido un material a procesar es más alta que la temperatura al que este se cristaliza, se denomina proceso de laminación en caliente, lo que significa que el material puede recristalizarse después del proceso de formación en caliente.

Este procedimiento es utilizado para grandes operaciones de desbaste que comienzan con lingotes de acero y no ferrosos. Este proceso de laminación se utiliza como una operación intermedia para reducir rápidamente el espesor del metal más cerca de su tamaño final (Figura 2.13).

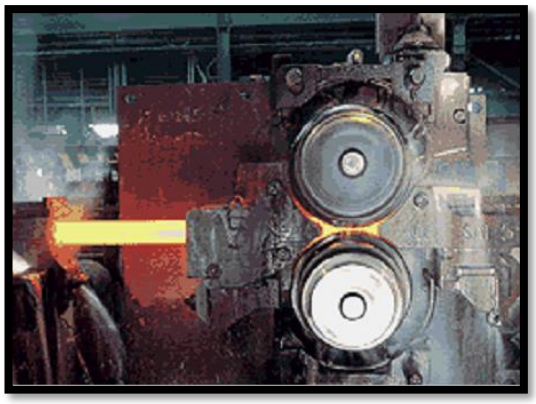

Figura 2. 13. Proceso de laminado en caliente [7].

2.2.4.2. Laminado en frío

Es un proceso de trabajo de metales en el que la temperatura de la pieza de trabajo es más baja que la temperatura de recristalización. Este tipo de laminación se utiliza como procedimiento de acabado de láminas y tiras donde se requiere un buen acabado superficial y tolerancias dimensionales estrechas. Además, el endurecimiento por deformación significa que los metales sometidos a este proceso de formación tienen mejores propiedades mecánicas. El proceso de laminación en frío tiene la misma base que el proceso de laminación en caliente, es decir, se aprovecha la elasticidad del acero para crear una deformación permanente; para ello, el material se alimenta entre dos cilindros, donde la distancia entre los dos cilindros es menor que el espesor de la cinta original [7].

2.2.5. Defectos en los productos laminados

Pueden surgir varios problemas durante la laminación en función de la interacción entre la deformación elástica generada en los molinos y rodillos, así como la deformación plástica que sufre el acero. A causa de la influencia de una gran fuerza en el proceso de laminación, el material rodante se dobla, y el laminador se deforma elásticamente. Las aberturas de los rollos deben ser perfectamente paralelas, de lo contrario, la disminución del

espesor no será constante a lo ancho y largo del material y algunas áreas de la hoja se hincharán con este valor constante.

Teniendo en cuenta los materiales utilizados, los tipos de defectos encontrados son diferentes (Fig. 2.14), y podemos clasificarlos de acuerdo a la forma, color, tamaño o ubicación en la bobina [8], por ejemplo:

Agujeros: Una abertura que penetra completamente al metal y afecta tanto la parte inferior como la superior de la hoja, suelen presentarse en conjunto con otros tipos de defectos en los bordes.

Impresiones: Se presentan como marcas con bordes claros que se distribuyen irregularmente en el material.

Cáscara: Es una sobreposición del material de forma irregular sobre la superficie de la lámina, el exceso no siempre es del mismo material y puede tener una apariencia en ocasiones oscura o brillante.

Rústico: Se hace presente sobre la superficie del material con apariencia de oxido o corrosión, por lo general tiene una tonalidad obscura. El tamaño y forma suelen variar.

Abolladura: Depresiones superficiales periódicas.

Arañazo: Se presentan como marcas de una longitud que varía en la dirección de bobinado, las cuales pueden revelar la ausencia de material.

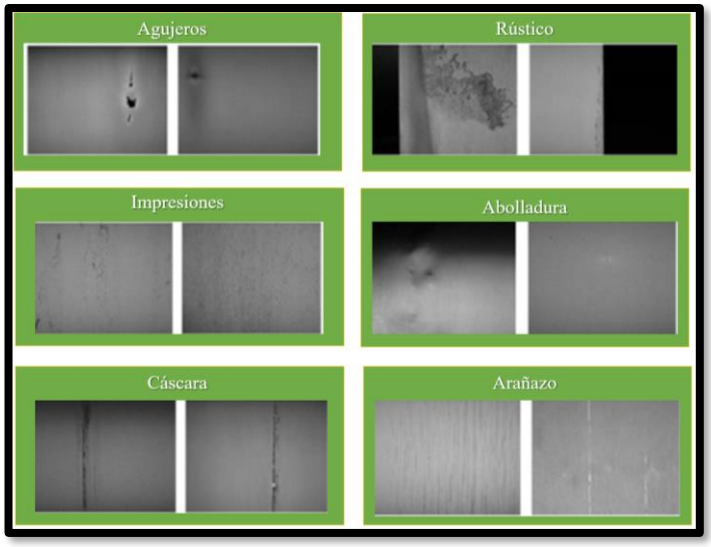

Figura 2. 14. Defectos comunes en una bobina metálica [9].

2.3. MÉTODOS DE INSPECCIÓN NO DESTRUCTIVOS

2.3.1. Definición

Un método no destructivo evalúa la calidad de un material sin la necesidad de modificar sus propiedades o estado natural, esto se logra por medio de diversas pruebas o ensayos las cuales permiten detectar y evaluar discontinuidades. Este tipo de ensayos está basado en principios físicos de los cuales, no se muestran de manera absoluta los resultados, por lo que deben tener una interpretación según las especificaciones del método utilizado [10].

Algunos de los objetivos que se espera tener con la realización de estos métodos son los siguientes:

- Prevenir accidentes.
- Garantizar la confiabilidad y calidad.
- Productos de calidad uniforme.
- Efectuar mantenimientos predictivos.
- Reducir gastos.

2.3.1.1. Tipos de métodos de inspección

De acuerdo con su tipo de aplicación, los métodos no destructivos comúnmente más utilizados para los ensayos se dividen en los siguientes:

2.3.1.2. Líquidos penetrantes

Este método permite ubicar defectos o grietas en la superficie del material basado en el principio de capilaridad, ya que permite que un líquido con baja viscosidad penetre en las hendiduras o grietas en la pieza a analizar [11]. Este tipo de prueba consiste en la colocación de un líquido de color fluorescente sobre la superficie de estudio, el cuál penetrara las discontinuidades que pudieran existir; después de cierto tiempo se retirara el exceso de líquido y se aplicará un revelador, el cual será absorbido por el líquido, revelando de esta manera alrededor de las discontinuidades [12].

2.3.1.3. Termografía

Este método se basa en seguir la evolución de la temperatura de un cuerpo durante o después de que energía radiactiva le sea suministrada, la cual permite que la visión del ser humano pueda prolongarse por el espectro infrarrojo [13]. Lo anterior permite adquirir una imagen térmica conocida como termograma, producto de la captura de emisiones naturales de radiación, en donde se puede observar la distribución técnica de todas las unidades del sistema, estableciendo la temperatura presente en cada punto de la superficie de manera instantánea y segura. Una de las cualidades más grandes de este método es la productividad, ya que esta inspección se puede realizar sin pérdidas o reducción de rendimiento, cuando el sistema esté en pleno funcionamiento.

2.3.1.4. Partículas magnéticas

Estas técnicas se basan en el magnetismo para la inspección. La aplicación consiste en magnetizar la pieza a inspeccionar, utilizando un polvo fino de partículas magnéticas como la limadura de hierro e induciendo corriente eléctrica para que, por medio de la visualización del campo magnético sea posible identificar alteraciones en la pieza: anormalidades, grietas o defectos superficiales, abolladuras, entre otras [11]. El imán permanente se puede utilizar pocas veces, ya que solo se puede conseguir campos magnéticos

débiles. Las partículas magnéticas pueden aplicarse en un ambiente seco o húmedo; pueden llegar a ser de diversos colores e incluso fluorescentes [12].

2.3.2. Inspección visual

La inspección visual consiste en un examen y/o evaluación a un sistema, en el cual se hace uso de los sentidos humanos apoyados de dispositivos los cuales magnifiquen la capacidad sensorial del inspector [11]. Es una técnica muy utilizada en muchas áreas de la industria, gracias a esta técnica es posible detectar grietas y defectos en el 80% de los productos. Este ensayo no requiere de un alto adiestramiento llevar a cabo la inspección de forma satisfactoria, sin embargo, el resultado de estas evaluaciones va directamente enfocado a la experiencia del inspector durante la ejecución de la inspección [12]. Se ha demostrado que su uso en la industria tiene una acción de prevención ya que es capaz de identificar problemas que pudieran verse reflejados en procesos posteriores de la fabricación del producto[14].

La técnica de inspección visual se puede dividir en dos grandes grupos:

2.3.2.1. Inspección visual directa

Se efectúa a una distancia relativamente corta del objeto, tomando ventaja de esta manera la capacidad visual natural de la persona que realiza la inspección. Es posible ampliar su alcance apoyándose de instrumentos como linternas, lámparas, lentes para aumento, algunos instrumentos de medición, las cuales ayudan a mejorar las condiciones de localización [12].

2.3.2.2. Inspección visual remota

Se emplea cuando el componente a inspeccionar no se encuentra de fácil acceso. En esta técnica es posible obtener fotografías o imágenes en movimiento con las cuales se podrá realizar un registro para análisis posteriores o cuando se necesite consultar alguna información pasada [12].

2.3.2.3. Aplicaciones

Al ser una técnica que necesita contar con mucha información para determinar el estado general de la pieza a inspeccionar, así como la detección de irregularidades las cuales puedan afectar el funcionamiento de los equipos, este tipo de análisis nos ayuda a demostrar que, al ser aplicada, puede detectar problemas los cuales podrían tener mayor magnitud en etapas subsecuentes proceso [14].

Obtención de resultados: Ya que este método se fundamenta en supervisar la calidad de las condiciones de la superficie durante la fabricación de los productos, generalmente se obtienen los siguientes puntos:

- Evaluación general de elementos tubulares, herramientas o componentes.
- Detección temprana de los defectos antes de alcanzar un tamaño crítico.
- Detección de defectos en manufactura.

Esta técnica es una de las primeras en considerarse al momento de llevar a cabo la inspección de un elemento con el fin de dar un dictamen del estado en el cual se encuentre la pieza a inspeccionar [15].

Procesos en los que se utiliza

Un ejemplo del uso de la inspección visual en la industria aeroespacial es en los propulsores de las aeronaves (turbinas de gas), ya que, para la inspección de su control de funcionamiento, se utiliza un endoscopio, este puede ser rígido o flexible, el cual obtendrá diversas imágenes las cuales ayudaran a la detección de grietas finas en las piezas críticas. Durante la inspección de las turbinas (impulsor, etapas de compresor, cámara de combustión, etc.) se analiza si los discos de turbina (blisks) corresponden a un giro de la turbina de gas por medio de endoscopios de vista lateral o cabezales de cámara totalmente giratorios (no es necesario utilizar espejos). Otro ejemplo es en la inspección de motores turbopropulsores y motores de hélice, ya que, en ambos, se analizan los pistones por medio de endoscopios, válvulas y transmisores [16].

2.3.2.4. Dispositivos utilizados

En la inspección visual es común el uso de dispositivos de ayuda y se pueden clasificar como [11]:

Espejos y sistemas de iluminación

Comprende los dispositivos que ayudan a identificar de manera más clara los defectos que se presentan en la superficie de los materiales mediante la utilización de la luz o un contraste, los cuales permiten tener un mejor manejo en la intensidad y el alcance de la misma por medio de espejos, pantallas filtros o el uso de luz polarizada.

Dispositivos de magnificación de imagen

Permiten aumentar la superficie bajo inspección con el uso de componentes ópticos, los cuales comprende la utilización de microscopios y lupas. Estos dispositivos pueden aumentar la superficie de inspección entre 5 y 10 veces su tamaño real, denominados microscopios, por otra parte, los microscopios son aquellos que el campo de amplificación puede ser entre 10 y 200 veces el tamaño real del objeto de interés.

Figura 2. 15. Dispositivo de magnificación de 40x [11].

Equipos de observación remota y endoscopios

Los endoscopios son artefactos los cuales permiten observar en áreas que por lo general son muy estrechas o de acceso difícil, normalmente este tipo de herramientas se

utiliza en superficies remotas o irregulares, sin verse en la necesidad de desarmar la pieza o equipo a analizar.

Estos tipos de dispositivos se dividen en tres categorías diferentes: sistemas de video, rígidos y flexibles. Los tubos rígidos van de 0,01 a 0,05 m de diámetro y constan de un sistema de lentes y espejos con opciones de aumento de 3x a 50x. Para iluminar el área a observar, tiene una pequeña bombilla incandescente cerca del campo de visión y tiene una luz externa conectada a una fibra óptica.

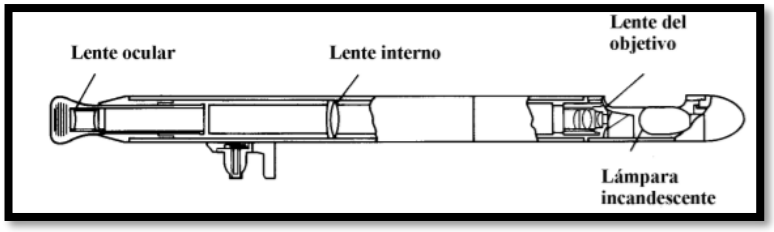

Figura 2. 16. Endoscopio rígido [11].

Los endoscopios flexibles surgen del desarrollo de las fibras ópticas, en las cuales una se utiliza para transmitir luz e iluminar el área de examen, y la otra para transmitir la imagen de observación. Debido a que la fibra actual es de tan alta calidad que no pierde resolución ni resistencia con la distancia, puede abarcar longitudes muy largas (hasta 30 m). Los endoscopios flexibles suelen utilizar sensores de imagen digital con monitores de vídeo.

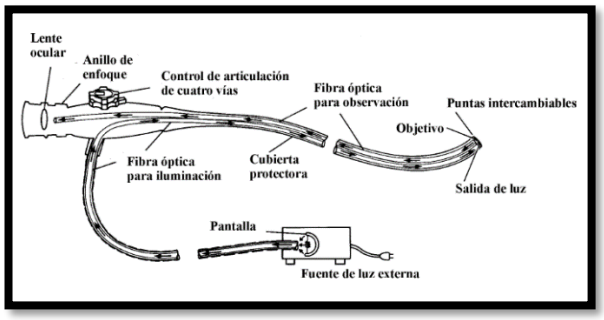

Figura 2. 17. Endoscopio flexible [11].

Una de las mayores ventajas de usar un endoscopio es la capacidad de observar áreas restringidas con acceso limitado y su uso en diversas condiciones y situaciones. Por otro lado, el principal inconveniente es la imposibilidad de definir la probabilidad de detección de

grietas, y es un método de inspección que lleva mucho tiempo si el área de interés es muy grande.

Sensores de imagen

Estos son dispositivos de video o fotografía que registran la investigación para la futura evidencia necesaria para un análisis posterior. Este tipo de imágenes generalmente se digitalizan porque a menudo se procesan para aumentar o mejorar su resolución, teniendo en cuenta aspectos de detalle que no se encuentran en el trabajo de campo original.

2.3.2.5. Procedimiento

Este proceso se puede realizar por medio de la utilización de espejo amplificadores, baroscopios y otros instrumentos de visualización, pero a continuación se presentará una explicación general de cómo se realiza este tipo de análisis [15]:

Calibración de equipo

El proceso se inicia calibrando el equipo ajustando el aumento, contraste y brillo correctos, con el fin de la mejor calidad de imagen y/o video de las superficies bajo inspección.

Selección de la superficie

Utilizando una sonda es posible explorar la pared interna de la pieza, la cual detectará discontinuidades.

Toma de imagen o video

Ya que se ubicó una discontinuidad, se procede a tomar la e imagen o el video con los ajustes adecuados.

Exportación de archivos

Con el uso de un puerto de memoria USB se exporta la imagen y/o video a un sistema inteligente (dispositivo móvil, computadora, etc.).

Análisis de resultados

Se analizarán los resultados obtenidos (sean imágenes o videos) para determinar si existe algún tipo de discontinuidad.

Resultados

Se fundamenta exclusivamente en los datos obtenidos y su interpretación debe ser de manera objetiva y concisa en una secuencia lógica.

2.3.2.6. Ventajas y limitaciones

Tabla 2- 1. Ventajas y limitaciones, Inspección visual [15].

VENTAJAS	LIMITACIONES
Método de bajo costo.	Solo se puede utilizar en discontinuidades superficiales. Se considera la limitación de la vista humana.
Es aplicable en múltiples etapas de un proceso de producción.	Se necesita una fuente de luz efectiva.
Se puede realizar de una manera rápida y sencilla.	Es necesario que la superficie inspeccionada sea accesible.
La geometría de la pieza no llega a ser gran problema al momento de la inspección ya que se cuenta con varios dispositivos de ayuda.	Se requiere el uso de personal experimentado y capacitado para llevar a cabo la inspección.

2.4. VISIÓN POR COMPUTADORA

La visión artificial se encuentra dentro de las áreas más emocionantes de la investigación en IA. El objeto de estudio se centra en la creación de sistemas informáticos que puedan ver y procesar un objeto tanto en video como imagen de la forma como lo hacen los humanos. La visión artificial era limitada hasta hace poco. Este campo ha tenido un crecimiento acelerado en los últimos años y ha podido vencer a los seres humanos en algunas labores. Hay dos factores que han llevado a este crecimiento.

Primero, por la cantidad de información que se ha generado para permitir el entrenamiento y la mejora de la visión artificial. Se cree que cada día se comparten en línea

más de mil 1800 millones de imágenes [17]. Por otra parte, el incremento de la potencia de cálculo permite el análisis de datos. No es difícil ver que este campo se ha beneficiado de la llegada de diseños de hardware más eficientes.

2.4.1. Problemas relacionados con la visión por computadora

La visión artificial da la oportunidad de realizar una gama muy amplia de tareas. Los problemas se deben en gran medida a nuestra falta de comprensión de cómo funciona la visión y el cerebro humano, todavía hay un largo trayecto por recorrer. El estudio se centra principalmente en las siguientes cuestiones:

- Clasificación de objetos. ¿A qué clase de objeto del conjunto de clases conocidas a priori pertenece la siguiente imagen?
- Reconocimiento: ¿Qué tipos de objetos se observan en la foto?
- Verificación: dada una clase de objeto, ¿es este un objeto en la imagen?
- Detección: Dada una secuencia de objetos a identificar a priori, ¿cuál es la ubicación de este objeto en la imagen?
- Segmentación: ¿En cierta imagen qué píxeles corresponden a un objeto?
- Reconocimiento: ¿En la imagen que objetos se encuentran y dónde están ubicados?

La siguiente imagen da un mejor panorama de estos problemas.

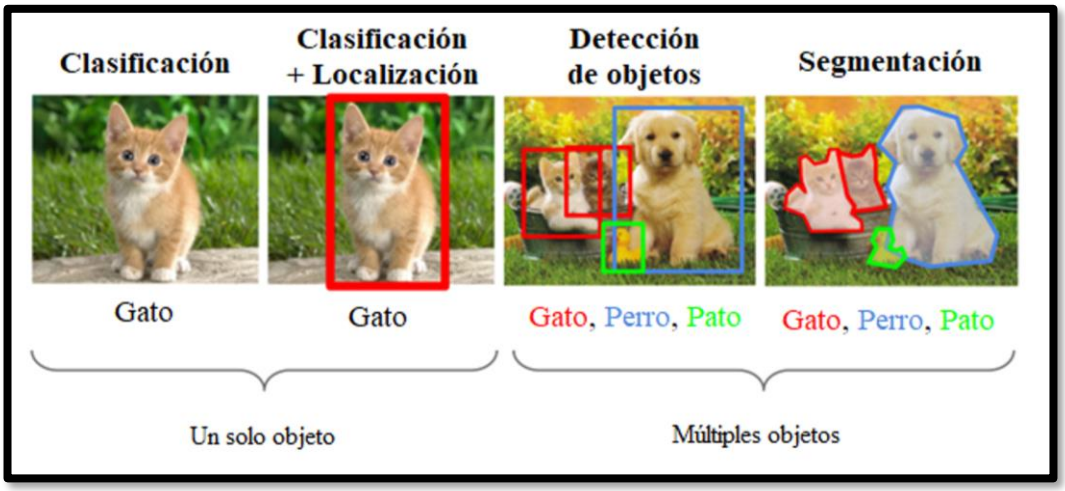

Figura 2. 18. Retos en la visión por computadora [18].

Este proyecto se enfoca en identificar un objeto contenido en una imagen, así como la ubicación de este en la misma.

2.5. APRENDIZAJE PROFUNDO

Una rama del aprendizaje automático que tiene como objetivo aprender rasgos de nivel alto en los conjuntos de datos se llama aprendizaje profundo [19]. Los modelos computacionales de varias capas pueden aprender y extraer información imitando la forma en que el cerebro procesa la información, lo que les permite percibir patrones subyacentes en grandes conjuntos de datos. El interés por esta técnica está relacionado con muchas áreas de interés de la inteligencia artificial, como el procesamiento del lenguaje natural [20], la visión artificial [21] [22], el aprendizaje por transferencia [23] [24]. La creación de modelos que imitan el cerebro humano condujo a la creación de redes neuronales. Estaban tratando de entender cómo el cerebro crea patrones muy complejos utilizando células de la misma familia. El modelo MCP, elaborado por McCulloh y Pitts, fue una parte importante en el desarrollo de las redes neuronales artificiales. La Figura 2.19 muestra qué contribuciones en este campo nos ayudaron a entrar en la era actual del aprendizaje profundo.

Año	Contribuyente	Contribución
300 A.C.	Aristóteles	Introdujo el Asociacionismo el cuál comenzó la historia del intento humano de entender el cerebro.
1873	Alexander Bain	Introdujo las agrupaciones neuronales como los primeros modelos de red neuronal, inspirado en la Regla de Aprendizaje Hebbiano.
1943	McCulloch & Pitts	Introdujeron el Modelo MCP, que se considera el antecesor del Modelo Neuronal Artificial.
1949	Donald Hebb	Considerado como el padre de las redes neuronales, introdujo la Regla de Aprendizaje Hebbiano que sienta las bases de las redes neuronales modernas.
1958	Frank Rosenblatt	Introdujo el primer perceptrón, que es muy parecido al perceptrón moderno.
1974	Paul Werbos	Introdujo la Retro propagación.
1980	Teuvo Kohonen	Presentó el Mapa de Autoorganización.
	Kunihiko Fukushima	Introdujo el Neocogitrón, que inspiró la Red Neuronal Convolucional.
1982	John Hopfield	Introdujo la Red Hopfield.
1985	Hilton & Sejnowski	Introdujeron la máquina de Boltzmann.
1986	Paul Smolensky	Introdujo el Harmonium, que posteriormente se conocería como Máquina de Boltzmann Restringida.
	Michael I. Jordan	Definió e introdujo la Red Neural Recurrente.
1990	Yann LeCun	Introdujo LeNet, que muestra la posibilidad de las redes neuronales profundas en la práctica.
1997	Schuster & Paliwal	Introdujeron la Red Neuronal Recurrente Bidireccional.
	Hochreiter & Schmidhuber	Introdujeron la LSTM, la cual solucionó el problema del gradiente de fuga en las redes neuronales recurrentes.
2006	Geoffrey Hinton	Introdujo las Redes de Creencia Profundas, también introdujo la técnica de preentrenamiento por capas, abrió la era actual del aprendizaje profundo.
2009	Salakhutdinov & Hinton	Introdujeron las Máquinas Profundas de Boltzmann.
2012	Geoffrey Hinton	Introdujo el Dropout, una forma eficiente de entrenar redes neuronales.

Figura 2. 19. Aspectos históricos importantes que han llevado a la era del aprendizaje profundo [25].

La era del aprendizaje profundo nació con la innovadora contribución de Hilton [26], quien introdujo las redes de creencias profundas en 2006. Esta red es la razón principal del crecimiento exponencial de los algoritmos de aprendizaje profundo en la última década. ¿Por qué las redes neuronales profundas se están volviendo más populares ahora? Hay muchos factores que contribuyen a la notoriedad de las redes neuronales profundas.

- Incremento en uso de sistemas de cómputo paralelo, multicore y multithread.
- Se ha incrementado la potencia de procesamiento debido al uso de unidades de procesamiento de gráficos.
- Existen numerosas plataformas de software que facilitan integrar fácilmente y a un nivel alto varias arquitecturas en el marco de computación de GPU sin entrar con detalles de bajo nivel.
- Se han introducido técnicas reguladas para prevenir el sobreajuste y mejorar el rendimiento a medida que escala.
- Avances en el diseño de algoritmos de aprendizaje que ofrecen buenas soluciones.

2.5.1. Redes neuronales convolucionales

La evolución de la visión por computadora y los avances en el aprendizaje profundo, se han logrado gracias a la red neuronal convolucional (CNN) [27].

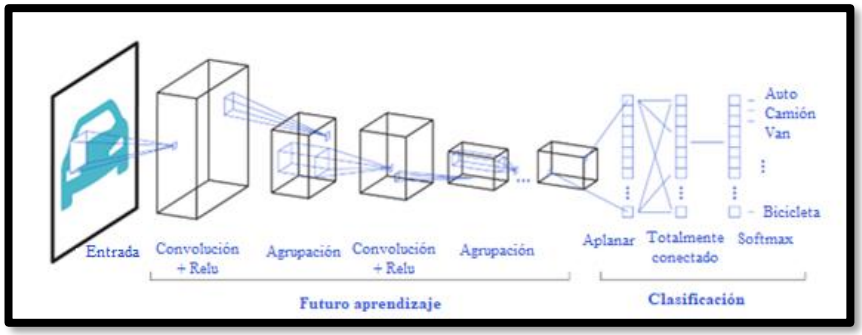

Figura 2. 20. Arquitectura de las redes convolucionales [28].

Se entiende por CNN a un algoritmo de aprendizaje profundo que en base a una imagen que tiene por entrada, se le es asignada una importancia, conocida como pesos del

aprendizaje, que son básicamente aspectos o características de la propia imagen lo que permite distinguirla de otras imágenes. CNN tiene por objeto replicar ciertos patrones de conexión neuronal en el cerebro humano y está fundamentada en la forma como la corteza visual está organizada [29]. La arquitectura logra un equilibrio entre la alta habilidad para aprender de las características destacadas de la imagen y su escalabilidad en conjuntos de datos masivos [30]. Para cumplir con estas expectativas, las CNN reducen las imágenes a una forma más manejable sin perder los rasgos clave requeridos para hacer buenas predicciones. Las capas de convolución se basan en operaciones de convolución de imágenes. En el caso de CNN, la imagen de entrada se enreda usando filtros (también llamados núcleos) que generan mapas de características. Específicamente, la convención se realiza deslizando los filtros en la imagen de entrada. Cada posición se completa mediante la multiplicación de la matriz y la suma de los resultados del gráfico de funciones. Este diagrama representa esta actividad:

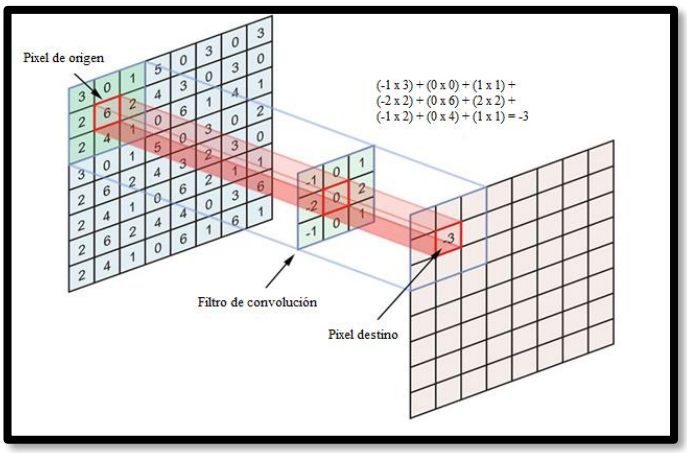

Figura 2. 21. Desplazamiento de filtro en la entrada y el resultado de la operación [31].

Las CNN no necesitan limitar la cantidad de capas convolucionales, por lo que generalmente se usan secuencialmente. La capa convolucional inicial es responsable de detectar características como bordes a un bajo nivel, colores, direcciones de degradado, etc. Las capas de agrupación son responsables de reducir el tamaño espacial de los elementos convolucionales. El propósito es tener una reducción el costo computacional que se necesita para el procesamiento de datos. Max Pooling retorna un valor máximo de la parte de la imagen procesada y Average Pooling retorna el valor medio la totalidad de los valores de la

parte bajo análisis. En términos generales Max Pooling promete un desempeño superior. El siguiente diagrama ilustra la actividad máxima de la operación descrita:

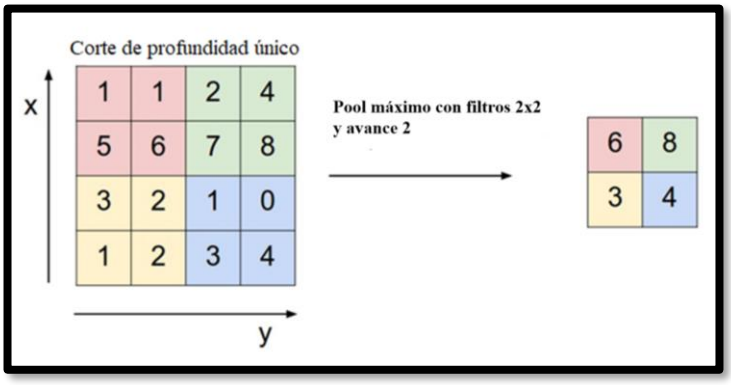

Figura 2. 22. Los valores más grandes se toman por Max pooling [31].

Lo complejo de la imagen puede requerir un incremento en el número de capas para percibir detalles a un nivel más bajo, lo que también aumenta en costo computacional.

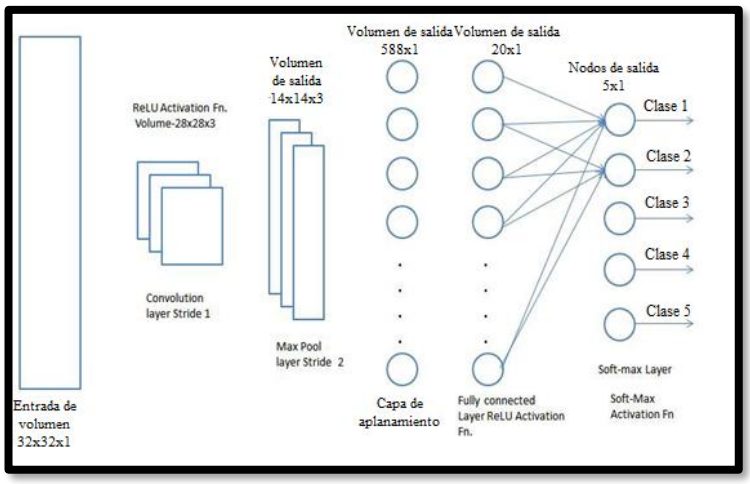

Figura 2. 23. Capa fully-connected y red neuronal feed-forward [28].

Después de aplicar estas capas secuencialmente, el modelo puede identificar características clave. El resultado de este proceso por lo general es la entrada a una capa totalmente conectada. Finalmente, permite que las salidas de las capas completamente conectadas se alimenten a una red neuronal de avance. Durante el entrenamiento continuo, la red neuronal aprenderá a diferenciar entre características principales y de bajo nivel de una

imagen para clasificar la imagen en la clase adecuada. Vale la pena mencionar que, aunque este trabajo intenta brindar una breve introducción a las CNN, hay muchas arquitecturas disponibles. Algunos de los más conocidos son: LeNet [32], AlexNet [21], GoogLeNet [33], ResNet [34].

2.5.2. Transferencia del aprendizaje

Se refiere a una técnica que nos permite afrontar los problemas más comunes cuando a aprendizaje profundo se refiere:

- Alto costo en reentrenamiento a causa de grandes volúmenes de datos.
- Enorme gasto de recursos informáticos para las labores del reentrenamiento.

Esta técnica se basa en el conocimiento del aprendizaje del modelo en conjuntos de datos para reducir el resentimiento. Esta idea se basa en un modelo de que el modelo reunió los datos anteriores que convirtieron el modelo en los nuevos datos para incrementar el proceso de preservación [35]. Con frecuencia, la transferencia del conocimiento se da de un modelo a mayor escala a uno de menor. En este caso, la red se entrena utilizando el conjunto de datos general COCO [36]. El set de datos se centra en la detección y ubicación de noventa clases de objetos, y utilizamos el aprendizaje por transferencia para incrementar el reentrenamiento del conjunto de datos de defectos superficiales que se usa en este proyecto. El siguiente diagrama muestra cómo ocurre el mencionado proceso.

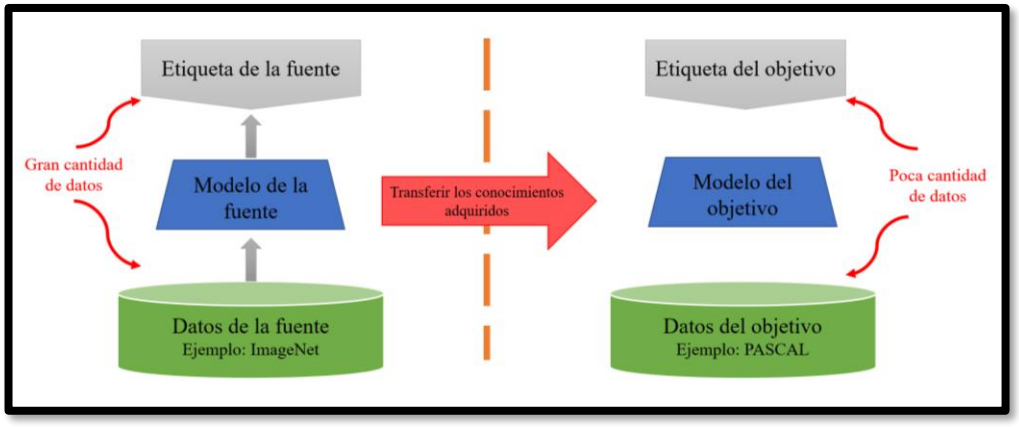

Figura 2. 24. Transferencia del aprendizaje [37].

Cuando se habla de una CNN, el aprendizaje de transferencia se logra inicializando la capa de captura de características en un nuevo conjunto de datos, tomando cuantificaciones de una capa de características. Más específicamente, los parámetros de las capas responsables de capturar objetos de bajo nivel, generalmente se usan porque están presentes en todos los datos. La detección de objetos implica múltiples capas de procesamiento después de la extracción de características porque proviene de un clasificador, porque no solo predice la clase del objeto, sino también su ubicación. Esto significa que volver a entrenar las redes es requiere un costo alto en términos de tiempo y procesamiento, motivo por el cual se evitan semanas de reentrenamiento de la red.

2.6. ARQUITECTURAS DE DETECCIÓN DE OBJETOS

Se pueden distinguir dos líneas de investigación en las arquitecturas de aprendizaje profundo en la actualidad, la primera de las cuales es anterior a la segunda. La primera, llamada arquitectura de dos etapas, primero genera regiones candidatas para ubicaciones de objetos en la imagen y luego clasifica los objetos asignándoles una clase como: rayones, inclusiones o perforaciones. En contraparte, un segundo detector es famoso por su arquitectura con una fase ya que realiza una mayor abstracción porque afrontan la problemática de detección de los objetos mediante regresión y clasificación. De esta manera, podrían en algún momento inferir la ubicación y clasificación de los objetos.

2.6.1. Arquitecturas de dos fases

La arquitectura de dos fases se conoce como arquitectura basada en regiones. Esto se remonta al año 2012, dado que AlexNet [21] ganara el ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), llevando las redes neuronales convolucionales al área del reconocimiento de objetos. La técnica para clasificación se lleva acabo usando una ventaja de deslizamiento de izquierda a derecha y de arriba a abajo. Un ejemplo es la siguiente imagen:

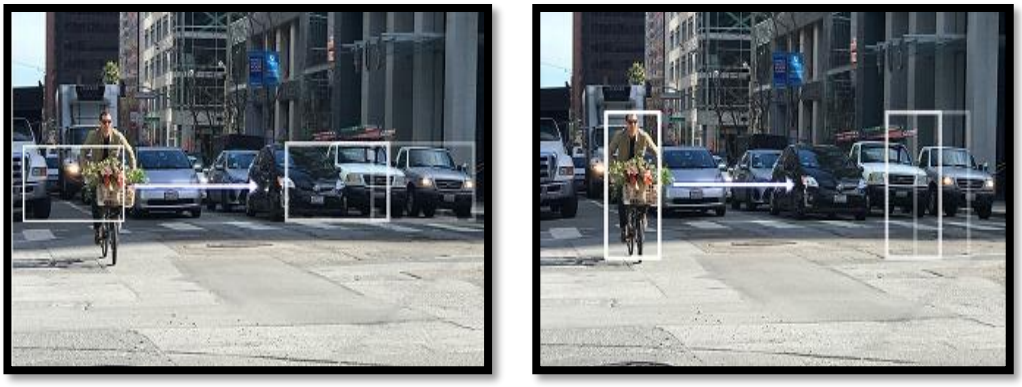

Figura 2. 25. Enfoque de ventana deslizante sobre la imagen [38].

En la Figura 2.25 podemos ver como necesitamos utilizar ventanas con diferentes formas y tamaños si queremos detectar diferentes objetos. En cada paso de desplazamiento, se toman secciones de la imagen de acuerdo con estas ventanas. Antes de la convolución del clasificador, el parche de alimentación de la red se ajusta a introducirse en la red. Esto no afecta la exactitud de la red porque está entrenada para procesar estas imágenes. La red es responsable de extraer una cierta cantidad de funciones de la sección. La imagen 2.26 muestra esta arquitectura:

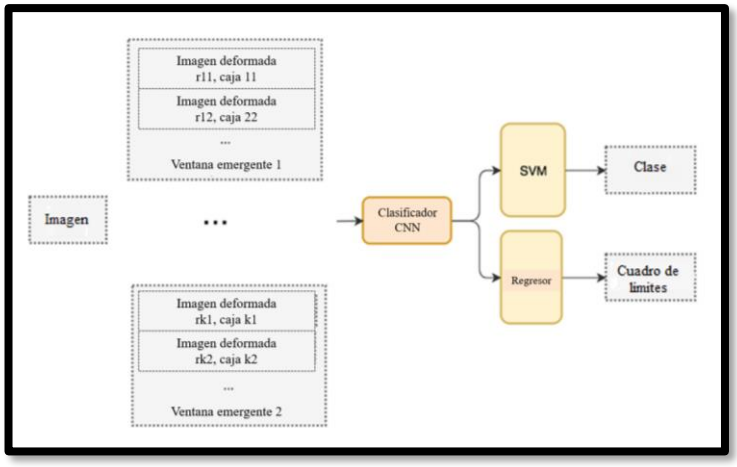

Figura 2. 26. Enfoque de fuerza bruta [38].

Aun así, los métodos de fuerza bruta son bastante costosos en términos de inferencia y tiempo de entrenamiento y son computacionalmente intensivos. Una mejora significativa es la reducción en la cantidad de secciones de arrastre. Al principio, cada píxel en sí pertenece

a un grupo. Luego se calcula la estructura de cada grupo y conecte una estructura similar. Para evitar que el grupo grande se coma al grupo pequeño, la asociación del grupo pequeño tiene prioridad sobre el grupo grande. El proceso continúa hasta que se cumple una cierta correlación. En la Figura 2.27, podemos ver cómo se forman estos grupos en la primera fila.

Figura 2. 27. Método de proposición de regiones: búsqueda selectiva [39].

Ross Girshick [40] en el 2013 propuso la primer arquitectura R-CNN utilizando el método de propuesta de región de interés. La arquitectura sugiere casi 2000 ROI por imagen. Luego, cada región se procesa por separado, usándola como entrada a una red neuronal convolucional. En la imagen 2.28 se muestra esta arquitectura:

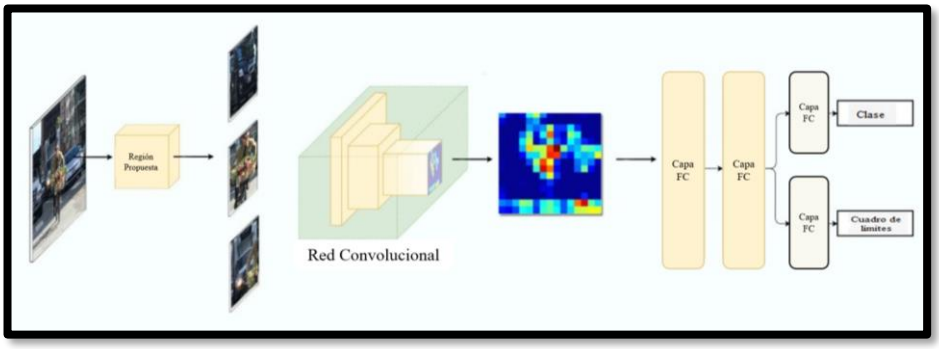

Figura 2. 28. R-CNN propuesta por Ross Girshick [38].

Un aspecto que destaca R-CNN sobre otros métodos es debido a su capacidad para identificar ROI de mayor calidad y, por lo tanto, mejorar la precisión del reconocimiento de objetos y una reducción de tiempo de ejecución. Aun así, R-CNN sigue siendo lento durante el

entrenamiento y la inferencia, principalmente porque presenta muchas regiones para ser preciso, y muchas se superponen. Al mismo tiempo, utilizamos un método de propuesta de región externa, como la búsqueda selectiva, con el fin de crear regiones de interés, que luego se ajustan con las características extraídas para formar parches. Estos parches se alimentan en capas totalmente conectadas que manejan la clasificación y ubicación de objetos. En la imagen 2.29 se muestra dicha arquitectura:

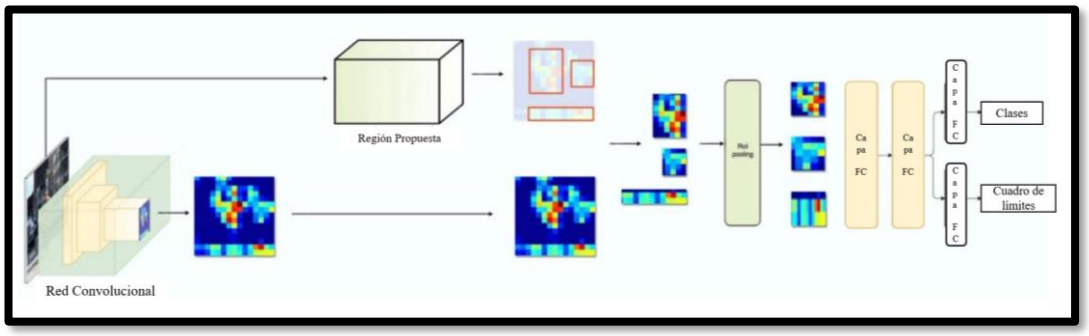

Figura 2. 29. Arquitectura Fast R-CNN [38].

En el año 2015, Ross Girshick [41] presento Fast R-CNN el cual reducía de forma considerable el tiempo en que se realizaban las detecciones en comparación con su predecesor R-CNN. Sin embargo, el detector segua siendo lento ya que buena parte del procesamiento se seguía realizando en la CPU, por lo que Shaoqig Ren propuso Faster R-CNN [42]. Esta propuesta de red recomienda regiones basadas en características derivadas de CNN y es más eficiente en la generación de ROI. En la Figura 2.30 se ilustra estar arquitectura.

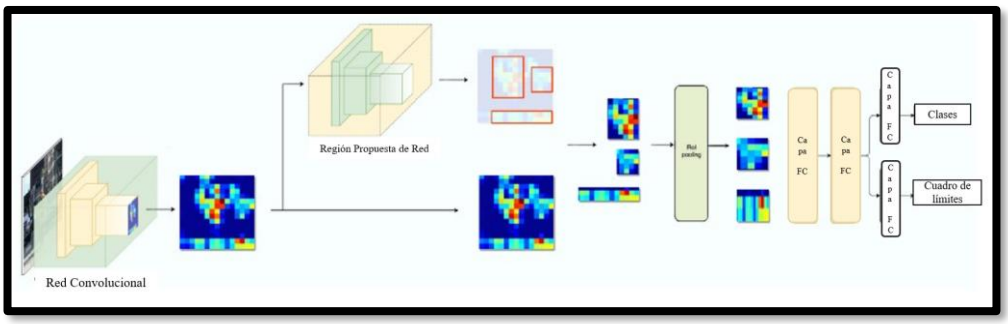

Figura 2. 30. Arquitectura Faster R-CNN [38].

Podemos ver algunas mejoras con esta mejora, excepto que el método de propuesta de región se reemplaza por RPN lo que hace a Faster R-CNN más rápido que su predecesor. Pero ahora en algunas cosas excede a R-CNN. Es muy costoso clasificar con una capa completamente atada. En su lugar, comienza a trabajar en capas de convolución, que son mucho más rápidas que las redes totalmente conectadas. Surgen redes nuevas como R-FCN: completamente basadas en regiones [33] y Resnet [36]. La integración de Resnet condujo a la red R-FCN [43] convertirse en un detector completamente convexo que supera con creces a Faster R-CNN en términos de precisión y rendimiento. En conclusión, hay algunas arquitecturas nuevas que van más allá de las ya implementadas.

2.6.2. Arquitectura de una fase

Este tipo de redes son de última generación para la localización y clasificación precisas de objetos, su inconveniente es que son demasiado lentas para el reconocimiento del objeto en tiempo real. Las arquitecturas de una sola etapa han surgido por la necesidad de lograr detectar objetos en tiempo real. Los expertos decidieron reducir el número de etapas e introducir una arquitectura en una etapa que es capaz de obtener la posición del cuadro que delimita el objeto y la probabilidad de que corresponda a la clase que intentamos identificar. A continuación, se presentará una arquitectura de un solo paso para la detección de objetos.

2.6.2.1. Detector de disparo único (SSD)

En 2015, SSD fue desarrollado con el propósito de detectar objetos en tiempo real [44]. La red tiene mejoras en términos de velocidad de procesamiento y detección, lo cual hace posible al eliminar la necesidad de utilizar una red de propuesta de regiones (RPN). También tiene otras mejoras como cajas estándar y capacidades de zoom múltiple. En otros problemas de detección, como el conjunto de datos COCO, presenta mejoras que elevan la precisión que proporciona Faster R-CNN utilizando imágenes con una baja resolución. Para que el proceso sea más fácil de entender, se muestra cómo SSD lleva acabo la detección de los objetos de predicción única. La Figura 2.31 muestra la arquitectura SSD con una predicción.

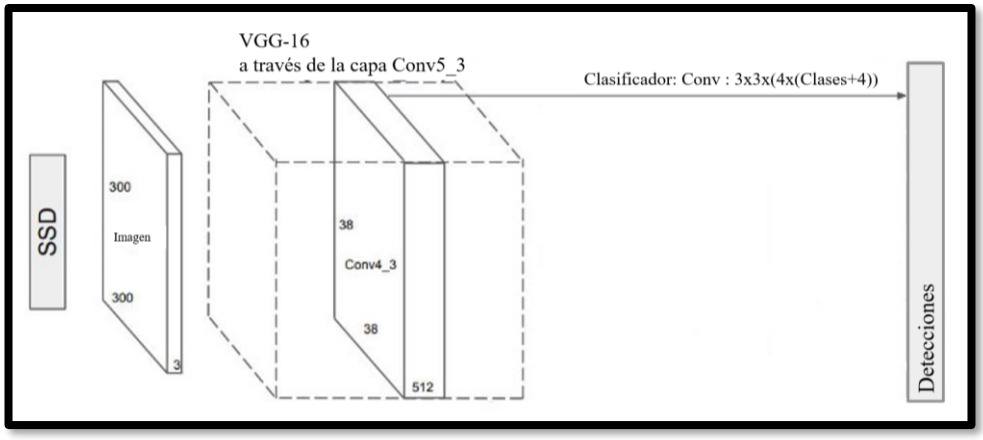

Figura 2. 31. SSD basado en un solo predictor [44].

Como se puede ver en la Figura 2.31, en primera instancia se lleva a cabo un cambio en el tamaño de entrada de la imagen de 300x300 pixeles. Esto ocasiona que la red haga inferencias a gran velocidad, pero a costa de no detectar objetos pequeños. Después de cambiar el tamaño, la extracción de la imagen se realiza mediante CNN, específicamente VGG16. De hecho, es una mejora en esta red convolucional, ya que inicialmente se utilizó VGG16 como clasificador de imágenes de alta calidad. Para hacer esto, cada cuadro en B predice los cuadros delimitadores, así como las probabilidades de que el cuadro tenga al objeto. Para exponer un ejemplo se utilizó una reducción de la capa Conv4_3 para dividir la imagen 8×8 para obtener:

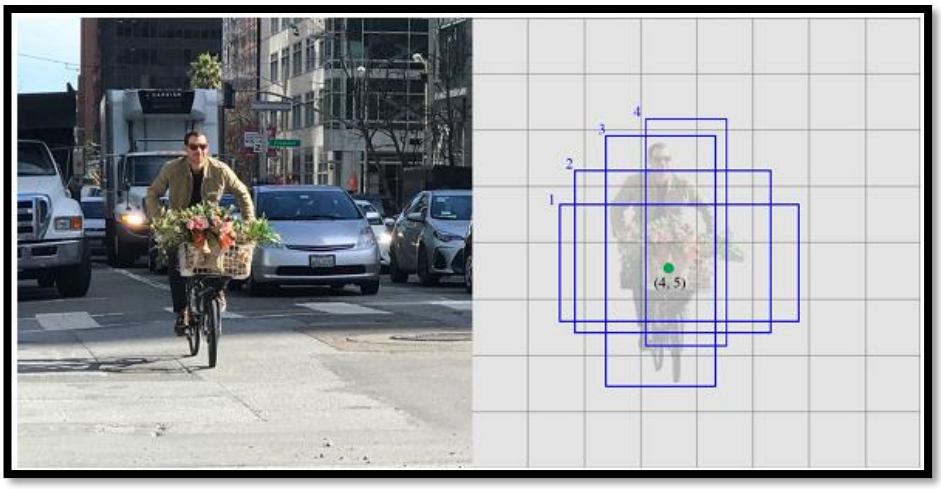

Figura 2. 32. Efectuando 4 predicciones por casilla utilizando capa Conv4_3 [45].

La detección consta de un cuadro delimitador y 4 puntos (uno para cada clase, en este caso ciclistas, peatones y coches, más una clase adicional si no se encuentran objetos). Posteriormente a la clase que tenga mayor puntaje se le asigna el valor de la predicción, en esta ocasión corresponde a la clase de bicicleta. En notorio que el costo de procesamiento dependerá de la cantidad de cajas que se encuentran en la imagen, la capa realizara las predicciones y en su mayoría no contendrán objetos para lo cual es necesario considerar una clase de objeto vacío para cuando se presente esta situación. La Figura 2.33 ilustra el resultado de esta operación:

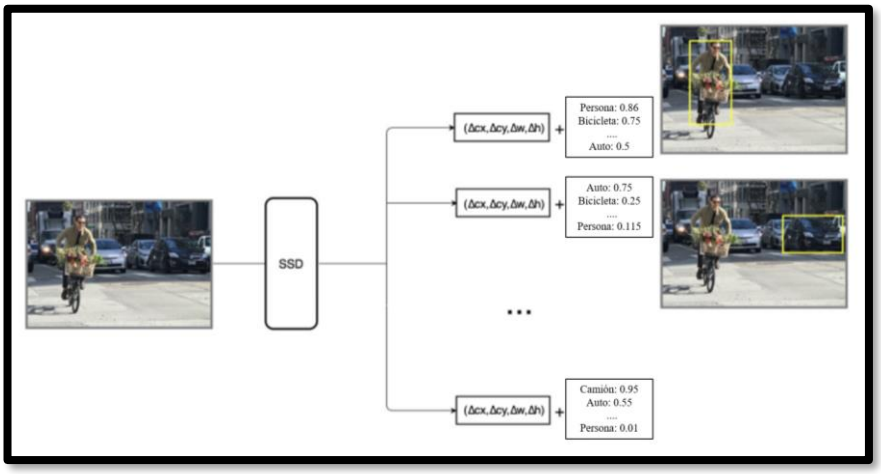

Figura 2. 33. Caja delimitadora para predicción (considerando el caso de que no exista objeto) [45].

La cantidad de recuadros en los cuales la imagen será dividida es una limitación importante de esta red, ya que, si los recuadros son muy grandes, estos no van a poder detectar algunos objetos muy pequeños porque dentro de cada recuadro contendrá más objetos. También se considera que al utilizar una estructura del tipo pirámide permitirá que la capa identifique objetos de diferentes tamaños, y cada capa está entrenada en este tamaño particular. Por ejemplo, se usa una capa 4 × 4 para objetos grandes y capas de objetos más pequeños 8 × 8:

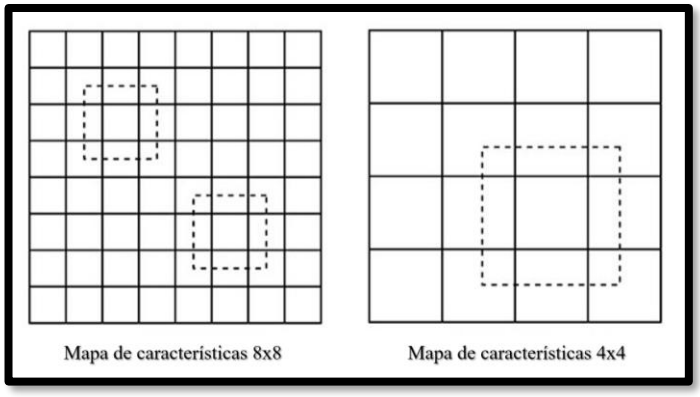

Figura 2. 34. Detección de objetos utilizando capas 4 × 4 y 8 × 8 [45].

A continuación, la Figura 2.35 ilustra como es la arquitectura SSD multi-detector:

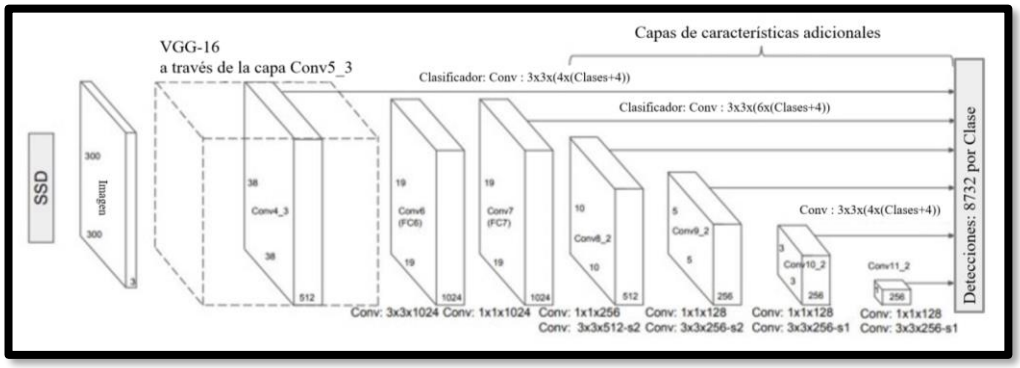

Figura 2. 35. Arquitectura SDD multi -detector [44].

La arquitectura SSD tiene en su totalidad las capas discutidas anteriormente. Primero, cambia el tamaño de la imagen a 300×300. Luego vinieron las aplicaciones de filtrado para detectar el objeto, por ejemplo, utilizando la capa Conv4_3. Posteriormente, la última capa en el lado derecho es la extracción. Otro detector basado en arquitectura SSD es SSD Mobile Lite [46]. Es una red entrenada en el conjunto de datos COCO, que está diseñado específicamente para entornos con recursos limitados. La ventaja de este detector es que requiere diez veces menos parámetros que YOLOV2, manteniendo la precisión alcanzada en SSD. Debido a estas propiedades, se sabe que es eficiente y, por lo tanto, un candidato ideal para la inferencia de detección en tiempo real de objetos.

2.6.2.2. YOLO (You Only Look Once)

En la actualizad, los sistemas para detección de objetos reutilizan clasificadores para llevar a cabo sus tareas, estos sistemas clasifican el objeto y posteriormente lo evalúan en diferentes ubicaciones, así como escalas. Con anterioridad se ha hablado de otras arquitecturas para la detección de objetos, pero YOLO [47] redefine la forma de realizar las detecciones basándose en el concepto de regresión simple. Esta novedosa técnica propuesta por YOLO permite que con solo una red convolucional sea capaz de predecir múltiples recuadros delimitadores, así como la probabilidad para cada clase en cada recuadro. Una ilustración de esta operación se puede ver en la Figura 2.36. YOLO optimiza su rendimiento para detectar objetos utilizando imágenes de forma completa, presenta varias ventajas en relación con otros métodos de detección:

- YOLO se caracteriza por su rapidez. Tratar las detecciones mediante una técnica de regresión, elimina la necesidad de sistemas complejos. La detección se puede predecir simplemente ejecutando una red neuronal en nuevas imágenes. Además, la precisión promedio de YOLO es muy superior comparado con otros detectores en tiempo real.

- YOLO utiliza imágenes completas. Comparado con los métodos de ventana deslizante y región, YOLO trabaja con la imagen completa a lo largo de la fase de entrenamiento y la evaluación, por lo que codifica internamente información contextual sobre la clase y su aspecto.

- YOLO es muy general. Debido a que Yolo aprende de los objetos generalizables, es poco probable que empeore cuando se use para un nuevo dominio o importaciones inesperadas.

Aun contando YOLO con ventajas importantes respecto a otras arquitecturas, presenta algunas dificultados cuando trata de ubicar ciertos objetos con precisión, por lo general esto se presenta con objetos muy pequeños.

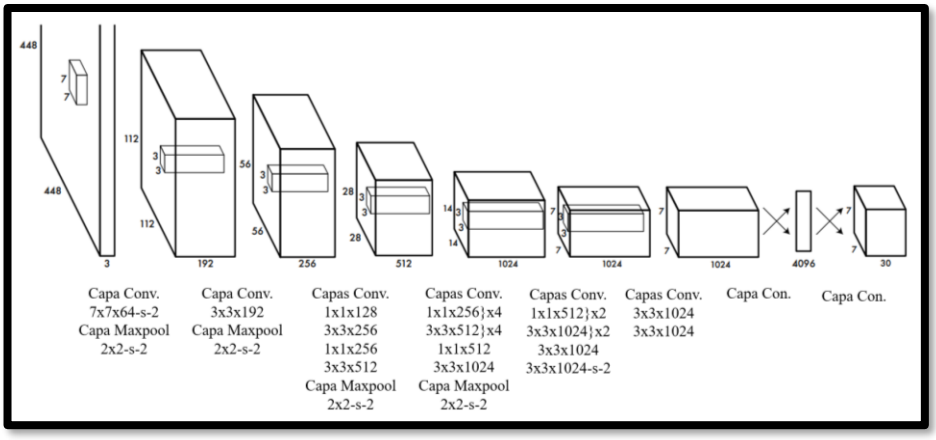

Figura 2. 36. Arquitectura YOLO en su primera versión, con 24 capas convolucionales [47].

YOLO ha lanzado algunas ediciones, comenzando con la compilación original, que podemos ver en la Figura 2.36 hasta el último YOLOv4 [48] lanzado el 23 de abril de 2020, usando YOLOv3 [49] y YOLOv3 Tiny [50]. Debido a su tamaño más pequeño, es más rápido, lo cual es una característica esencial para detectar defectos en la superficie de un material, cuando se trata de lograr un procesamiento en tiempo real y, por lo tanto, recibirá una atención especial.

Funcionamiento general de YOLO

La idea detrás del marco YOLO es combinar distintas tecnologías de detección en una sola arquitectura. Su función general se describe a continuación:

1. Primero, divide la imagen en cuadrículas de dimensión CxC. Si el objeto de interés se encuentra ubicado en la parte central del recuadro, es responsabilidad de la celda el identificar el objeto.

2. La celda responsable registra los cuadros delimitadores de B y su confianza, lo que representa la creencia que en el interior del recuadro hay un objeto y que el cuadro realmente corresponde al objeto encontrado, lo que se representa como *Confianza = P (Objeto)* $* IOU \frac{truth}{pred}$. En el caso que el objeto no se encuentre en esa celda, por consecuencia la confianza será de cero.

3. Los recuadros delimitadores se conforman de 5 parámetros: *x, y, w, h* y la confianza.

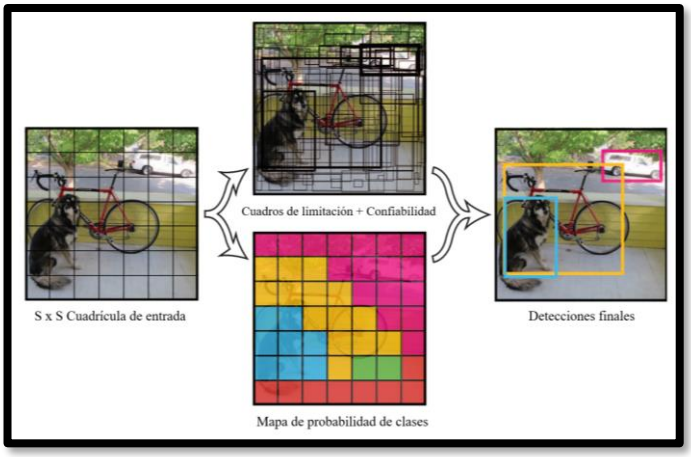

Figura 2. 37. Funcionamiento general de YOLO [47].

Finalmente, la probabilidad de cada clase se multiplica por la probabilidad de cada casilla de detección, lo que da como resultado la probabilidad específica de cada casilla de detección para cada clase:

$$P\,(\,Clase\ i\ |\ Objeto\,)\ *\ P\,(\,Objeto\,)\ *\ IOU\,\frac{truth}{pred} = P\,(\,Clase\ i\,)\ *\ IOU\,\frac{truth}{pred} \quad \textbf{(Ec. 2- 1)}$$

Esta métrica muestra la probabilidad para la cual dicha clase podría aparecer en el recuadro detectado.

2.7. MÉTRICAS UTILIZADAS EN LOS DETECTORES DE OBJETOS

A continuación, se da una explicación de las métricas que son ampliamente utilizadas para la detección de objetos, la más importante es mAP (*mean average precisión)* [51].

2.7.1. Precisión y Recall

Precisión: se encarga de medir que tan exactas son las predicciones que realiza el modelo utilizando un porcentaje de predicciones correctas. La representación matemática es [51]:

$$Precision\ = \frac{TP}{TP + FP} \qquad \textbf{(Ec. 2- 2)}$$

Para lo cual TP indica un verdadero positivo (*True Positive)* y FP un falso positivo (*False Positive).*

Recall: conocida también como sensibilidad, es la responsable de medir la habilidad para realizar detecciones de casos positivos, dicho de otra forma, los objetos que se encuentran presentes en alguna de las categorías del set de datos. Matemáticamente se puede expresar de la siguiente manera [51]:

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}} \qquad \textbf{(Ec. 2- 3)}$$

Donde FN indica un falso negativo (*False Negative)*. Concretamente, ***recall*** es el porcentaje de casos positivos que fueron acertados correctamente.

2.7.2. IOU

Intersección sobre Unión o IOU es la responsable de medir el área de intersección entre dos cajas delimitadoras. La representación matemática es [51]:

$$\boldsymbol{IOU} = \frac{\boldsymbol{\text{área de intersección}}}{\boldsymbol{\text{área total de la unión}}} \qquad \textbf{(Ec. 2- 4)}$$

Esta medición nos será de utilidad para identificar si la predicción que se efectuó puede ser considerada como buena. Generalmente es necesario establecer un umbral para considerar si la detección se considera aceptable o no, considerando como validas todas las predicciones que superen un 0,7. En la Figura 2.38 se puede tener una mejor apreciación de estos conceptos, especialmente la conocida como verdad básica (*ground truth)* ya que indica la intersección que ocurre entre la caja delimitadora de la predicción y la caja real que fue etiquetada en la imagen [57].

Figura 2. 38. Intersección sobre unión [51].

2.7.3. AP y mAP

Precisión promedio (*average precisión)* o AP puede ser representado en una gráfica, si la precisión la asignamos al eje *y*, de la misma forma asignamos recall al eje *x*, entonces AP sería el área por debajo de la curva. En la Figura 2.39 es posible identificar esta área [51].

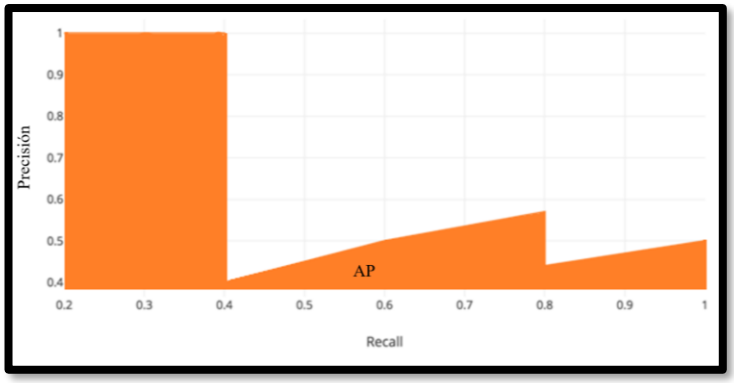

Figura 2. 39. Gráfica de precisión y recall [51].

La definición matemática más conocida es:

$$\boldsymbol{AP = \int_0^1 p(r)dr} \qquad \textbf{(Ec. 2- 5)}$$

Para lo cual, *p(r)* se refiere a la curva generada por precisión y recall. Por lo tanto, podemos definir a mAP como la media de la precisión promedio (AP) en un umbral dado por IOU. El

umbral para IOU generalmente es [0.5 , 0.05 , 0.95], por lo que podemos considerar a mAP como la media de AP para cada valor de IOU [51].

2.7.4. Índice F1 (F1 Score)

En una métrica ampliamente utilizada en los modelos de detección ya que resume la precisión y sensibilidad (recall) en un solo valor. Es muy útil cuando la distribución de las clases es desigual, por ejemplo, cuando el número de observaciones de la clase que no contiene rasgos a identificar es del 15% y las de la clase de interés es del otro 85% restante. El F1-Score se calcula como la media armónica entre precisión y sensibilidad. Es una métrica equilibrada entre tener pocos falsos positivos y pocos falsos negativos [52]:

$$\boldsymbol{Indice\ F1} = \frac{\boldsymbol{2 * Presición * Recall}}{\boldsymbol{Presición + Recall}} \quad \textbf{(Ec. 2- 6)}$$

2.7.5. Matriz de confusión

La matriz de confusión es una herramienta base para conocer y evaluar que tan efectivo es el algoritmo o modelo de detección utilizado para resolver el problema de identificación de defectos. En particular, sirve para mostrar de forma explícita cuándo una clase es confundida con otra. Se llama "matriz de confusión" porque hace que sea fácil detectar dónde el sistema está confundiendo dos clases [52].

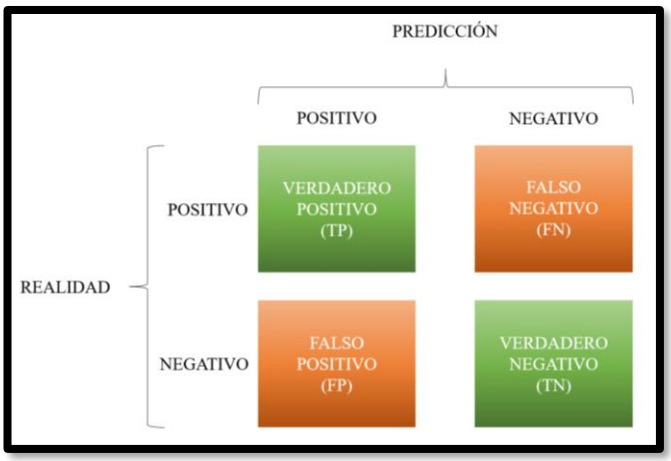

Figura 2. 40. Matriz de confusión [52].

- Verdadero Positivo (TP): cuando la clase real es 1 (Verdadero) y la predicha es también 1 (Verdadero).
- Verdaderos Negativos (TN): cuando la clase real es 0 (Falso) y el pronosticado también es 0 (Falso).
- Falso Positivo (FP): cuando la clase real era 0 (False) y el pronosticado es 1 (True).
- Falso Negativo (FN): Cuando la clase real era 1 (Verdadero) y el valor predicho es 0 (Falso).

Las métricas precisión, sensibilidad (recall) e índice F1 antes mencionadas se pueden obtener de la matriz de confusión, por tanto, podemos identificar cuatro casos posibles al analizar el comportamiento de sus valores con respecto a la clase que se desea identificar [52]:

- Alta precisión y alto recall: el modelo maneja perfectamente esa clase. Se trata de un buen modelo de detección para poderse implementar.
- Alta precisión y bajo recall: el modelo no detecta la clase objetivo muy bien, pero cuando lo hace es altamente confiable.
- Baja precisión y alto recall: El modelo detecta bien la clase objetivo, pero también incluye muestras de la/s otra/s clase/s.
- Baja precisión y bajo recall: El modelo no logra identificar la clase correctamente, por lo que no sería recomendable seguir adelante con la implementación de dicho modelo.

CAPÍTULO 3
MARCO TEÓRICO

En esta sección se analizan una por una las diversas tecnologías que se pueden utilizar en el desarrollo de aplicaciones móviles y las razones para elegir una u otra. En parte se exponen los sistemas operativos para plataformas móviles más populares, así como su respectiva plataforma de desarrollo.

3.1. CLASIFICACIÓN DE LAS APLICACIONES MÓVILES

Hay tres enfoques que se deben evaluar cuando se habla de desarrollo de aplicaciones móviles: aplicaciones nativas y multiplataforma (web e híbrido). En esta sección se dará una breve explicación de cada una de ellas.

3.1.1. Aplicaciones nativas

Son diseñadas para poderse ejecutar en una plataforma específica, por lo que se debe tener en consideración el dispositivo en el cual se ejecutará, el sistema operativo soportado, la versión y detalles específicos aunados a la plataforma. Cuando se finaliza el desarrollo de la aplicación, esta puede ser publicada en las tiendas de aplicaciones digitales para su distribución, previamente antes de su publicación deben de pasar por un proceso de revisión para validar que cumple con lo indicado para la distribución y pueda ser accesible para el usuario. La ventaja que ofrece el uso de estas aplicaciones radica básicamente en que se puede tener acceso e interactuar con todas las funciones del dispositivo como son la cámara digital, GPS, por mencionar algunos.

Un punto importante es que no requieren de una conexión a internet permanente para poder ser utilizadas. Se inician rápidamente, pueden ejecutarse en segundo plano y pueden en enviar notificaciones al usuario para avisarle que ocurre un evento que requiere su atención. Estas ventajas naturalmente vienen con mayores costos de desarrollo, ya que se deben usar diferentes lenguajes de programación según la plataforma. Por lo tanto, si desea abarcar varias plataformas, debe generar una aplicación para cada plataforma.

Algunas ventajas se enlistan a continuación:

- Capacidad de interactuar con la totalidad de funciones de hardware del equipo.

- No requieren una conexión permanente a internet.
- Rápida ejecución.
- Enviar notificaciones a los usuarios.
- Mejor experiencia de usuario.

Algunas desventajas son:

- Cada plataforma requiere tiene su propio lenguaje de programación.
- Por lo general su desarrollo incurre en cosos de altos.

3.1.2. Aplicaciones web

Las aplicaciones web móviles están diseñadas para ejecutarse en un navegador móvil mediante una URL, que por un lado tiene la ventaja de que no requieren instalación. Estas aplicaciones se desarrollan utilizando lenguajes como HTML, CSS y JavaScript, las mismas tecnologías que se utilizan para crear sitios web. Son independientes al sistema operativo, por lo que pueden ejecutarse desde varias plataformas.

La principal ventaja es que no requieren una instalación como tal, ya que la aplicación propiamente se ejecuta desde Internet, pueden ser ejecutadas desde cualquier plataforma por su independencia con el sistema operativo, solo se requiere contar con un navegador y acceso a internet. Por otro lado, su ejecución es lenta y puede volverse menos atractivo que las aplicaciones nativas. Además, pueden tener un rendimiento deficiente debido a problemas de conectividad. Otro punto importante a considerar es que dichas aplicaciones tienen no tienen acceso a los recursos de hardware del dispositivo donde son ejecutadas.

Algunas ventajas son:

- Se desarrollan más fácilmente y pueden ejecutarse en cualquier plataforma.
- La publicación no requiere aprobación.
- Los usuarios siempre tienen la última versión de la aplicación.
- Se ha desarrollado una "respuesta de red".

Las desventajas que presentan son:

- No cuentan con acceso a las funciones de hardware del equipo.

- Para utilizarlo, además de pagar la transmisión de datos, que no siempre es posible para los usuarios, se requiere una conexión a Internet.
- La experiencia grafica del usuario no es la mejor ya que los contenidos no se ajustan perfectamente.

3.1.3. Aplicaciones hibridas

Estas aplicaciones comparten aspectos importantes de los tipos de aplicaciones mencionados anteriormente. Se desarrollan utilizando tecnologías multiplataforma, pero hay disponibles muchas características específicas del dispositivo. En resumen, se desarrollan utilizando tecnologías web y se ejecutan en un contenedor web en dispositivos móviles.

Las principales ventajas de este enfoque incluyen la capacidad de distribuir las aplicaciones a través de la tienda de aplicaciones, ejecución en múltiples plataformas y la ventaja de aprovechar las capacidades de hardware del equipo. Un inconveniente es que todas las plataformas usan la misma interfaz y las aplicaciones no parecen aplicaciones nativas.

Algunas ventajas son:

- Disponibilidad en tiendas de aplicaciones virtuales.
- Aunque están desarrollados en tecnología multiplataforma, deben instalarse como aplicaciones nativas.
- La misma base de código se utiliza en varias plataformas.
- Tiene acceso a la mayoría del hardware del equipo.

En menor medida, también presentan algunas desventajas:

- La experiencia que tiene el usuario es más parecida a una página web en lugar a una aplicación propia del equipo.
- La interfaz de usuario no presenta mucha similitud con el aspecto del sistema operativo.

Elección

La finalidad del proyecto es desarrollar una aplicación la cual tenga la capacidad de procesar imágenes y video en tiempo real para la detección de defectos, por lo que se requiere acceder a los recursos de hardware del dispositivo, particularmente, la cámara integrada, para aprovechar al máximo las capacidades técnicas del equipo con el fin de obtener un mayor desempeño y velocidad de procesamiento.

Por otra parte, considerando las condiciones de operación y asumiendo que el dispositivo deberá ser montado en un punto estratégico a lo largo de una línea de producción en la industria, no es posible garantizar la disponibilidad a una red de comunicación o el acceso a internet, motivo por el cual la aplicación no deberá depender de estos recursos para poder funcionar, siendo capaz de operar de manera autónoma.

En términos de desempeño, al buscar alcanzar la mayor velocidad de procesamiento y disponibilidad durante el proceso, una aplicación web o híbrida presentaría un problema de latencia (tiempo que le tarda en transmitir los datos en una red), ya que, para lograr las tareas concernientes a la detección de defectos, estos enfoques necesitarían forzosamente depender de la disponibilidad y estabilidad de los recursos de la red.

Considerando los aspectos antes mencionados, es recomendable desarrollar una aplicación nativa, ya que de esta manera garantizamos la disponibilidad de la misma en todo momento, eliminando la necesidad de mantener una conexión a internet activa, sumando a ellos que la experiencia que el usuario tiene en las aplicaciones nativas es más placentera ya que gráficamente son más atractivas que en los otros tipos de aplicaciones, sumando a ello que es posible aprovechar al máximo los recursos del hardware, por estas razones se decidió por desarrollar una aplicación nativa para cumplir con este propósito.

3.2. SISTEMAS OPERATIVOS MÓVILES

Los sistemas operativos móviles están diseñados específicamente para dispositivos móviles como teléfonos inteligentes, asistentes digitales personales (PDA), tabletas u otros sistemas móviles integrados. Entre los sistemas operativos más utilizados podemos nombrar

a Android, iOS y Windows Mobile.

Sus principales funciones son:

- Administrar el hardware.
- Gestionar los recursos del equipo para evadir procedimientos contradictorios.
- Organizar el almacenamiento de archivos y carpetas en el equipo.

La Figura 3.1 muestra una gráfica donde se puede observar un estudio del Grupo Gartner [53], en el que puede observar la evolución de estos sistemas operativos en el mercado de acuerdo al número de dispositivos en los puntos de venta. Destaca la desaparición Nokia Symbian, el declive de BlackBerry, el lento crecimiento de Windows y la consolidación de Apple con alrededor del 15%. El gráfico muestra cómo Apple ha logrado un crecimiento significativo en las ventas al lanzar nuevos dispositivos cada año. Finalmente, cabe destacar el importante crecimiento de la plataforma del sistema operativo Android de Google, que ha alcanzado más del 80% de cuota de mercado en cinco años.

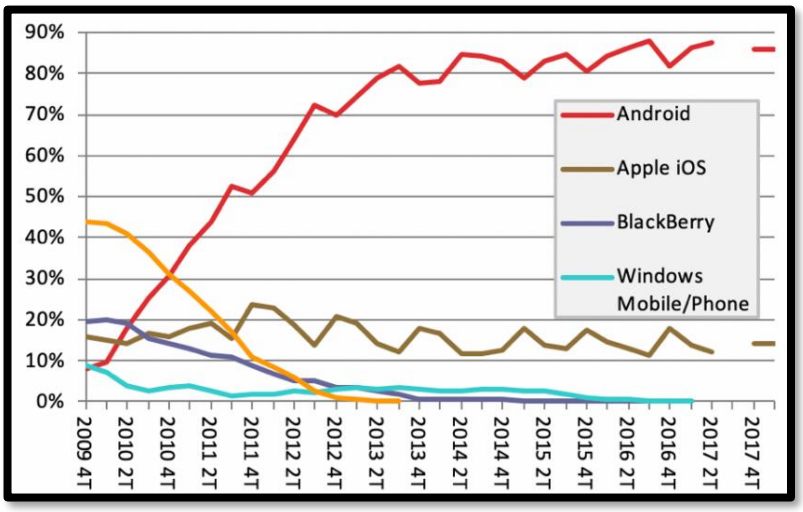

Figura 3. 1. Dispositivos vendidos mundialmente al primer trimestre de 2018, de acuerdo al SO (fuente: Gartner Group) [53].

En el siguiente apartado se explica las características más sobresalientes de los SO móviles.

3.2.1. Android

El sistema operativo Android fue diseñado para ejecutarse en equipos móviles como en dispositivos para el hogar u oficina, actualmente podemos encontrar televisiones, relojes, tabletas, inclusive vehículos que utilizan este sistema operativo con gran fluidez. Es un desarrollo basado en software de código abierto, como Linux Kernel desarrollado por Google, lo que lo ha popular por su uso sencillo y la gran cantidad de aplicaciones.

Historia y evolución de Android

La empresa que creó Android tiene su propio nombre, Android Inc., fundada en 2003 por Andrew Rubin, Rich Miner, Nick Sears y Chris White. Anunciaron Android como el sistema operativo original para cámaras digitales y les permitió conectarse a computadoras sin cables. Pero como el tema no era muy prometedor, eligieron los teléfonos móviles. En apenas dos años, Google se interesó por la empresa y decidió comprar Android por 50 millones de dólares para involucrar a los cuatro fundadores en la empresa. Sin embargo, habrá que esperar hasta la segunda mitad de 2008 para empezar a ver dispositivos con esta versión del sistema. En el segundo y tercer trimestre de 2010, Android alcanzó una cuota de mercado del 43,6 % en el segundo y tercer trimestre de 2010. Estados Unidos de América, en el cuarto trimestre de 2011, Apple iOS superó el poder de más del 50% en todo el mundo. A principios de 2018, había más de 2 millones de aplicaciones disponibles en la tienda oficial de aplicaciones de Android en Google Play. Incluso hemos visto que otras tiendas de aplicaciones no oficiales tienen toneladas de aplicaciones para el sistema operativo de Google [54].

Arquitectura de Android

Android está desarrollado en Java, C++, C y XML. Tiene una arquitectura de varias capas, como se indica en el artículo de los desarrolladores de Android "Arquitectura de la plataforma" [55]. A continuación, se hace una breve descripción de cada uno de ellos:

- Aplicaciones: Este nivel incluye aplicaciones que vienen incluidas con Android por defecto (Teléfono, Calendario, Navegador, Contactos, etc.), así como aplicaciones añadidas por el usuario posteriormente. Todas estas aplicaciones utilizan bibliotecas, API y servicios de nivel inferior. Por lo general, estas aplicaciones se desarrollan en

Java usando el SDK de Android, aunque se pueden desarrollar en C/C++ usando el NDK de Android.

- Gestor de Notificaciones: Permite que las aplicaciones notifiquen a los usuarios los eventos que se produzcan durante su ejecución.
- Administrador de paquetes: Le permite obtener información relacionada a los programas instalados en el equipo.
- Gestor de recursos: Administrar el acceso a los recursos.
- Administrador del teléfono: administre las funciones del teléfono.
- Sistema de visualización: proporciona una gran cantidad de elementos para crear una interfaz de usuario (GUI).
- Administrador de ventanas: administra las ventanas del programa.
- Servicio XMPP: permite utilizar este protocolo XML para intercambiar mensajes.
- Android Runtime consta de bibliotecas centrales, que son bibliotecas con una gran cantidad de clases Java y máquinas virtuales. Debido a que los dispositivos Android tienen poca memoria y procesadores limitados, no se puede cargar una máquina virtual Java estándar.
- Kernel de Linux: están contenidos los controladores para utilizar los componentes de hardware con las llamadas correspondientes. Se basa en Linux y proporciona gestión de memoria, procesos, batería o instalación de controladores para usar hardware, etc.

Ventajas e inconvenientes de Android

Entre las ventajas más destacables podemos encontrar:

- Es un sistema multiplataforma, lo que permite ejecutarlo en una amplia gama de dispositivos.
- Es de uso gratuito.
- Multitarea: puede hibernar los programas no utilizados y cerrarlos si no se han utilizado durante un tiempo.
- Montones de aplicaciones, la mayoría de las cuales son de uso gratuito.
- Actualizaciones del sistema operativo. Siempre que el dispositivo sea compatible con estas nuevas versiones, las nuevas versiones de Android aparecerán gradualmente, lo

que permitirá a los usuarios actualizar continuamente su sistema operativo.

- Interfaz intuitiva.

De igual forma, *Android* también tiene una serie de desventajas:

- Vulnerabilidad: dado que su naturaleza es de software libre, es más vulnerable a los ataques.
- En términos de multitarea, esto se convierte en un punto en contra, debido a que ejecutar múltiples aplicaciones en segundo plano conlleva a una ralentización del dispositivo, aumentando el consumo de batería.

3.2.2. iOS

IOS es un sistema operativo que inicia y usa Apple. Se derivó del sistema operativo de iPhone OS. Inicialmente se lanzó para los teléfonos esta marca, aunque con el paso de los años también se utilizó en otros equipos de la empresa, como los reproductores de música iPod o tabletas iPad. La principal diferencia con Android es que el sistema operativo de Google se puede instalar en innumerables marcas de teléfonos móviles, mientras que iOS es un sistema propietario cerrado para dispositivos de la misma marca.

Al igual que otros sistemas operativos móviles, iOS nos permite instalar apps que añaden funcionalidad a las que vienen de serie en nuestros smartphones. En otras palabras, además de llamadas o mensajes, puedes visitar la App Store para buscar funciones que se ajusten a tus necesidades, aprender inglés o realizar compras [56].

Historia y evolución de iOS

El primer iPhone se lanzó en 2007 y fue diseñado para impactar el mundo de la telefonía móvil, con una pantalla táctil capacitiva LCD de 3,5 pulgadas que podía conectarse a WiFi o auriculares, y tres opciones de almacenamiento de 4 GB, 8 GB o 16 GB. Pero hasta ahora no había App Store. Lanzado en junio de 2008, el iPhone comenzó a surgir con algunas diferencias, ya que el 2008 solo se tuvieron que descargar y transferir 500 aplicaciones originales por año. En 2009 esa cantidad se multiplicó por 10.000, ya estaban disponibles

50.000 apps. Para enero del 2007, Apple anunció los primeros avances hacia iOS, "iPhone OS", sin dar más detalles.

Se necesitaron algunos años para que el sistema operativo se lanzara sin problemas y el 29 de junio de 2009, se lanzó iPhone OS. La llegada del iPhone 4 y el iPad unos meses después en 2010 marcó otro paso adelante tal y como lo conocemos: Steve Jobs subió al escenario para mostrar su nuevo teléfono [56].

Arquitectura de iOS

La arquitectura de iOS se basa en capas, donde las capas superiores contienen los servicios y tecnologías más importantes para el desarrollo de aplicaciones, y las capas inferiores gestionan los servicios más básicos.

- Capa del kernel del sistema operativo.
- Capa de servicios básicos.
- Capa intermedia.
- Capa de "Cocoa Touch".

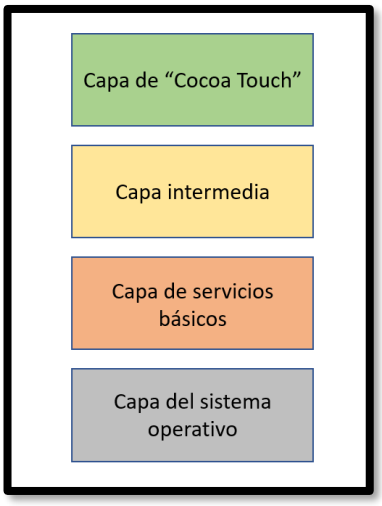

Figura 3. 2. Capas de iOS [57].

1. Cocoa Touch

La capa Cocoa Touch es la interfaz entre el dispositivo y el usuario. Esta capa es una interfaz única con un diseño único que consta de aplicaciones comprobadas que comparten

muchos de los modos que se encuentran en las Mac, pero se han rediseñado con un enfoque en las interfaces optimizadas y sensibles al tacto. La mayoría de estas aplicaciones se basan en Objective C [58].

2. Capa Intermedia

También conocida como la "capa de medios", es una interfaz mixta de C y Objective C que le permite realizar tareas o administrar archivos multimedia. Esta capa contiene tecnologías de gráficos, audio y video diseñadas para crear la mejor experiencia multimedia en dispositivos móviles [58].

3. Capa de servicios básicos

La capa de servicios básicos contiene los servicios del sistema central de la aplicación. El principal de estos servicios es Foundation que define los tipos básicos utilizados por todas las aplicaciones. Esta capa también contiene tecnologías separadas que admiten funciones como ubicación, iCloud, redes sociales y redes [58].

4. Capa del kernel del sistema operativo

La capa del kernel incluye la funcionalidad de bajo nivel de la que dependen la mayoría de las demás tecnologías. Incluso si no usa estos métodos directamente en su aplicación, es probable que otros sistemas lo hagan. En situaciones en las que necesite abordar explícitamente la seguridad o comunicarse con accesorios de hardware externos, puede usar el marco de trabajo de esta capa [58].

Ventajas y desventajas en iOS

Las ventajas más significativas son:

- Aplicaciones de muy buena calidad, ya que pasan por muchos filtros manuales antes de ingresar a la App Store.
- Optimización de los recursos del terminal, mayor duración de la batería.
- Capacidad para sincronizar toda la familia de productos de la marca Apple.
- Presenta al usuario una interfaz de fácil manejo.

Por otra parte, tiene algunas desventajas:

- Los precios de los equipos son elevados en contraste con otros fabricantes que ofrecen características técnicas similares a un menor precio.
- Para poder desarrollar para esta plataforma es necesario utilizar el entorno de desarrollo Xcode el cual es propiedad de Apple, además de necesitar una computadora Mac OS X regístrate mediante una cuenta con Apple para tener acceso a los servicios que ofrece.
- La programación es poco intuitiva y de complicado manejo.
- Como desarrollador, no te permite cambiar los componentes de su API lo que reduce la libertad a la hora de querer desarrollar nuevas características.

3.2.3. Windows 10 Mobile

Windows 10 Mobile es el sucesor de Windows Phone, el sistema operativo móvil de próxima generación desarrollado por Microsoft en respuesta al iPhone y Android. Se anunció durante el Mobile World Congress 2010 y se lanzó en noviembre de ese año. Se lanzó junto con diez dispositivos de marcas como HTC, Dell, Samsung y LG. Su interfaz gráfica difiere de iOS y Android, que eran copias del sistema operativo de BlackBerry durante su desarrollo inicial, pero con el éxito del iPhone, cambiaron de rumbo para parecerse lo más posible al iPhone. Con el tiempo, Windows 10 Mobile se instaló solo en los teléfonos inteligentes Nokia, y la división desapareció unos años más tarde cuando Microsoft dejó de fabricar teléfonos y la línea Lumia fracasó. El último dispositivo que se lanzó, el Lumia 650, fue hace unos años, en febrero de 2016 [59].

Historia y evolución de Windows 10 Mobile

En 1996, Microsoft introdujo un sistema operativo enfocado a dispositivos móviles con capacidades limitadas, disponible bajo el nombre de Windows CE. El sistema siguió evolucionando durante los años siguientes hasta Windows Phone 7 y sus predecesores Windows Phone 7.5, Windows Phone 7.8 y los últimos sistemas operativos basados en Windows CE. En 2003, lanzaron un sistema operativo móvil dedicado: Windows Mobile 2003, la primera versión lanzada bajo el nombre de Windows Mobile. Este sistema operativo fue seguido por Windows Mobile 5.0 en 2005, Windows Mobile 6.0 en 2007 y Windows

Phone 7 en 2010. El sistema operativo Windows Phone y los sistemas operativos comerciales anteriores estaban dirigidos al mercado de consumo y, por lo tanto, compiten con Android e iOS. En 2015, Microsoft presentó Windows 10 Mobile, que une todos los sistemas operativos. El sistema funciona en todo tipo de plataformas, ya sean smartphones, tabletas u ordenadores. En enero de 2019, Microsoft anunció que Windows 10 Mobile finalizará su ciclo de vida el 10 de diciembre de 2019, después de lo cual dejará de publicar actualizaciones de seguridad relacionadas con el sistema operativo y los servicios en línea, como las copias de seguridad. equipo) [60]. Sin embargo, Microsoft movió discretamente la fecha al 14 de enero de 2020 (para que coincidiera con la fecha de retiro de Windows 7) y lanzó actualizaciones de seguridad adicionales [59].

Arquitectura de Windows Phone

Windows Phone es el predecesor de Windows 10 Mobile y su arquitectura es completamente diferente al sistema operativo anterior. Tal y como señala Carlos Luis Murillo en la publicación “Windows Phone 8” [61], su arquitectura es la siguiente:

- Runtime: La interfaz del usuario está desarrollada en código C# y sigue el patrón sandbox para desarrollar fácilmente aplicaciones seguras.
- Herramientas: Las tecnologías que necesita para el desarrollo de aplicaciones son: el entorno de desarrollo Visual Studio, Expression Blend para el diseño de GUI y el emulador de Windows Phone que hace posible depurar y probar las aplicaciones.

Elección

Una vez estudiados los sistemas operativos que se mencionaron anteriormente, no se tienen ventajas muy claras con respecto a sus competidores, todos tienen sus ventajas y sus desventajas.

En el caso particular del sistema operativo Windows 10 Mobile, ofrece buena interactividad con el entorno de trabajo Windows 10, ampliamente utilizado en ambientes laborales, sin embargo, en su versión para dispositivos móviles, su presencia en el mercado es mínima comparado con sus competidores Android y iOS, sumando a ello, Microsoft, la empresa creadora de este sistema, dejo de dar soporte a esta plataforma a inicios del año 2020, dejando este sistema operativo en desuso, lo que no lo hace atractivo para el desarrollo

del proyecto, por las limitaciones y falta de compatibilidad con los dispositivos y sistemas recientes.

Relacionado al sistema operativo iOS, es posible crear aplicaciones de calidad, con una interfaz gráfica intuitiva y una gestión de recursos, sin embargo, presenta algunas limitaciones, por ejemplo, para desarrollar en iOS es necesario contar con un equipo Mac y un dispositivo móvil iPhone, ambos a precio poco accesible, esto debido a que el objetivo del fabricante es crear una experiencia de trabajo con dispositivos de la misma marca Apple, lo que suele ser un inconveniente al momento de desarrollar una aplicación, por la limitada compatibilidad con dispositivos de otros fabricantes y restricciones de interoperabilidad con librerías de software y aplicaciones creadas por terceros, sumando a ello, los numerosos filtros a los que las aplicaciones son sometida antes de poder estar disponible para los usuarios finales. Una de las principales motivaciones del proyecto es el uso de tecnologías fácilmente disponibles y de uso gratuito. Tal cual se mencionó con anterioridad, Android es el sistema operativo móvil más utilizado en la actualidad.

Otro punto importante, como se mencionó anteriormente, es que desarrollar iOS requiere una terminal Mac y un iPhone, los cuales son costosos. Por otra parte, desarrollar una aplicación para dispositivos Android requiere únicamente de un dispositivo informático con las especificaciones mínimas requeridas para instalar un entorno de trabajo, y cualquier dispositivo móvil con sistema operativo Android, que es mucho menos costoso el precio. Más accesibles en el mercado, lo que aumenta la posibilidad de ejecutar aplicaciones en dispositivos de diferentes fabricantes, dotando a los dispositivos de características de alto rendimiento, lo cual es un punto clave en el desarrollo de aplicaciones que realizan múltiples operaciones al mismo tiempo, las cuales se buscan lograr resultados inmediatos, porque la detección debe hacerse en tiempo real.

Finalmente, Android por ser una plataforma que está fundamentada en el desarrollo libre, se caracteriza por su amplia compatibilidad con herramientas, bibliotecas y recursos creados por terceros, permitiendo tener mayor control y flexibilidad, además de un ilimitado acceso a información por parte de comunidades de desarrolladores en todo el mundo. Dicho lo anterior, se concluye que Android es el sistema operativo seleccionado para el desarrollo de la aplicación para este proyecto.

3.3 ENTORNOS INTEGRADOS DE DESARROLLO PARA ANDROID

Esta sección presentará brevemente los entornos de desarrollo integrados (IDE) disponibles para el desarrollo de una aplicación Android y proporcionará una justificación para elegir el entorno más adecuado.

3.3.1. Eclipse

Es un entorno de desarrollo gratuito con licencia de código abierto desarrollado mediante un modo de actualización basado en complementos. Está diseñado para trabajar con varios lenguajes de programación. Sus características más destacadas son:

- Múltiples vistas y perspectivas lo que permite trabajar en varias perspectivas y ventanas.
- Colección de complementos: existe un buen número de complementos que permiten personalizar el entorno.
- Usar ANT para crear paquetes apk los proyectos creados en Android Studio no deben importarse.
- Para trabajar con Android se descarga un complemento auxiliar.

3.3.2. NetBeans

De la misma forma que Eclipse, NetBeans es gratuito y de licencia de código abierto. Funciona con múltiples lenguajes de programación. Sus características son:

- Asistentes y Project Managers: en NetBeans el desarrollo está basado en proyectos. Tiene un asistente que configura diferentes proyectos y elegir marcos. También cuanta con un depurador de código que permite monitorear en tiempo real valores de variables y propiedades.
- Manejo de bases de datos: le permite establecer una conexión a varios administradores de bases de datos. Usar ANT para crear paquetes apk por lo que los proyectos creados en Android Studio no deben importarse.
- Para trabajar con Android se descarga un complemento auxiliar.

3.3.3. IntelliJ IDEA

IntelliJ IDEA tiene dos distribuciones, una es IntelliJ IDEA Community Edition gratuita y de código abierto y la otra es IntelliJ IDEA Ultimate de pago. Los aspectos más destacados de la versión gratuita incluyen:

- Editor de código: le permite resaltar inmediatamente las advertencias y los errores. También es compatible con la finalización inteligente de códigos y la finalización automática sofisticada y receptiva.
- Herramientas de Android integradas: tiene un diseño de interfaz de usuario que le permite arrastrar y soltar elementos.
- Mayor productividad: compatibilidad con Maven y Gradle y control de versiones compatible con Git, GitHub, SVN y más.

3.3.4. Android Studio

El desarrollo de aplicaciones para el sistema operativo Android está muy extendido y es asequible (requiere un pago único de 25 dólares) como parte de la plataforma de desarrollo [62]. Por lo tanto, la empresa decidió lanzar un IDE oficial para desarrollar aplicaciones de Android, su software está disponible gratuitamente en el sitio web, así como sus herramientas para todos los usuarios. En comparación de Eclipse y NetBeans, fue diseñado específicamente para programar aplicaciones para dispositivos Android, por lo que no es necesario agregar ningún complemento para trabajar con Android [62]. Sus principales características son:

- Gestión de proyectos: Android Studio está basado en proyectos.
- Acceso a bases de datos: proporciona acceso al sistema de la base de datos y consulta las tablas y los datos almacenados en el sistema.
- Editor de interfaz de usuario: muestra en el archivo XML una vista previa con los cambios realizados.
- Le permite crear nuevos módulos en un solo proyecto sin tener que cambiar el espacio de trabajo para administrar el proyecto como es común en Eclipse.
- Crea paquetes .apk con gradle: le permite compilar desde la línea de comandos y crear fácilmente diferentes versiones de la misma aplicación. Le permite importar

proyectos creados en Eclipse o NetBeans.

- Al descargar Android Studio ya tendrás todas las herramientas que necesitas para desarrollar las aplicaciones.

Elección

Al revisar las características de cada uno de los entornos de desarrollo antes mencionados, se puede identificar que tanto Eclipse como NetBeans son muy similares en sus características y fáciles de instalar y usar, por otra parte, se cuenta con IntelliJ IDEA que parece ser una buena alternativa, solo que la versión gratuita presenta ciertas limitaciones que pudieran dificultar en cierta medida el desarrollo de la aplicación. De igual forma se puede identificar que algunas de las funcionalidades de Android Studio son menores que las de sus competidores, pero como se mencionó con anterioridad, también presenta ventajas sobre los otros IDE, destacando el hecho de que es el entorno oficial para desarrollar aplicaciones de Android, así que después de lo comentado anteriormente, se concluye que la aplicación se desarrollaría en Android Studio.

3.4 FRAMEWORK PARA ENTRENAMIENTO DE MODELOS

En esta sección se presenta brevemente el framework utilizado tanto para el entrenamiento de los modelos como para llevar a cabo las labores de inferencia en el dispositivo.

3.4.1. Tensorflow

Tensor Flow se caracteriza por ser una plataforma de código abierto o marco de aprendizaje automático integral que tiene todo un ecosistema de herramientas completo y flexible, además de poner a disposición de los usuarios recursos comunitarios y bibliotecas que permiten a los desarrolladores fomentar de manera innovadora el aprendizaje automático además de crear e implementar fácilmente aplicaciones basadas en el aprendizaje automático. Desarrollado originalmente por el equipo de Google Brain, Tensorflow nos brinda varios recursos para facilitar la creación, el entrenamiento y la evaluación del modelo automático de aprendizaje, con las siguientes características [63].

- Set de datos y modelos entrenados previamente desarrollados por la comunidad y por Google.
- Herramientas: ayuda a los usuarios a usar el ecosistema TensorFlow.
- Librerías: extensiones y librerías basadas en TensorFlow.
- Certificación de TensorFlow.
- Learn Machine Learning (Aprendizaje automático): un recurso educativo para familiarizarse con los conceptos iniciales del aprendizaje automático de TensorFlow.

3.4.1.1. Object Detector API

Los detectores de objetos pueden identificar cuál de un conjunto conocido de objetos podría estar presente y proporcionar información sobre sus posiciones dentro de la imagen dada o un flujo de video. Un detector de objetos está entrenado para detectar la presencia y ubicación de múltiples clases de objetos. Es posible utilizar la API ObjectDetector de la biblioteca de tareas para implementar detectores de objetos personalizados o preentrenados en las aplicaciones móviles.

Funciones clave de la API de ObjectDetector:

- Procesamiento de imágenes de entrada, incluida la rotación, el cambio de tamaño y la conversión del espacio de color.
- Localización del mapa de etiquetas.
- Umbral de puntuación para filtrar resultados.
- Resultados de detección Top-k.
- Etiqueta la lista de permitidos y la lista de denegados.

En este caso, la API de detección de objetos se utilizará para entrenar un modelo proporcionado por Tensorflow o una red neuronal convolucional preentrenada en la suite genérica de COCO. Estos modelos se pueden encontrar en el zoológico de modelos de detección de objetos de Tensorflow.

- CenterNet MobileNet V2 COCO
- EfficientDet D0 COCO
- SSD MobileNet V2 COCO
- Faster R-CNN Inception V2 COCO

3.4.1.2. Tensorboard

Tensorflow nos proporciona la herramienta Tensorboard, la cual brinda las herramientas y visualizaciones requeridas para la experimentación con el aprendizaje automático, tales como:

- Seguimiento y visualización de métricas como pérdidas y precisión.
- Visualización del gráfico de las capas y operaciones del modelo.
- Revelar datos de imagen, texto y audio.

El uso de esta herramienta monitorea el aprendizaje de nuestra red, ya que podremos ver la función de pérdida, la precisión y algunas imágenes que muestran las entradas que realiza nuestra red en cada iteración.

3.5 HERRAMIENTAS DE APOYO

Las siguientes herramientas son de gran utilidad para tareas relacionadas al aprendizaje automático y a continuación se da una descripción de cada una de ellas.

3.5.1 Google Colab

Colab es una herramienta desarrollada por el Google Research que permite escribir código Python y ejecutarlo desde cualquier navegador. Es particularmente útil para aprendizaje automático, análisis de datos y tareas educativas. Desde una perspectiva más técnica, Colab es una extensión de notebook Jupyter alojado que se ejecuta a través de la nube y le permite ejecutar y guardar código en Python. Es un proyecto de código abierto basado en Colab, no necesita configuración y proporciona de forma gratuita acceso para recursos informáticos como las GPU [64].

Colab también admite que los usuarios puedan compartir sus cuadernos de Jupyter Notebook con los demás usuarios. El código es ejecutado sobre una máquina virtual asociada a la cuenta del usuario. Las máquinas virtuales son eliminadas después de un período de inactividad y el servicio de Colab garantiza su vida útil máxima.

La vida útil máxima de una máquina virtual es de 12 horas. Además, la computadora portátil se desconectará de la máquina virtual si está inactiva durante demasiado tiempo. La vida útil máxima de una máquina virtual y el comportamiento del tiempo de espera inactivo pueden cambiar con el tiempo o según el uso. Estas limitaciones son necesarias con el fin de que Colab pueda proporcionar recursos informáticos de forma gratuita. El uso de esta herramienta fue elemental para desarrollar el proyecto, dado que brindo acceso gratuito a una máquina virtual con capacidades avanzadas de GPU para entrenar modelos y realizar ciertos procesos, con todo el procesamiento en la nube, sin necesidad de configuración adicional. El entorno está listo para usar, con Python y muchas bibliotecas instaladas de forma predeterminada.

3.5.2 Roboflow

Roboflow es una plataforma que permite etiquetar imágenes en conjuntos de datos de manera rápida e intuitiva y ayuda a generar conjuntos de datos en un formato que otras plataformas, como TensorFlow, pueden leer. Está ideado para hacer más sencillo el proceso de creación de modelos de visión artificial. Gracias a Roboflow, fue posible etiquetar las imágenes utilizadas para este estudio.

3.5.3. Anaconda

Anaconda forma parte de las distribuciones del lenguaje de programación Python que está enfocado a la comunidad científica (análisis predictivo, aplicación en aprendizaje automático, ciencia de los datos procesamiento a gran escala, por mencionar algunos). Tiene la ventaja de eficientar la administración e implementación de paquetes. Las distribuciones incluyen paquetes de ciencia de datos para Linux, macOS y Windows. Existen varias alternativas, por una parte, está la Anaconda Distribution o Anaconda Individual Edition que son distribuciones de acceso libre para uso personal y por otra parte cuenta con productos orientados a las empresas como Anaconda Team Edition y Anaconda Enterprise Edition, para propósitos comerciales que incurren en un costo [65]. Suele utilizarse para desarrollar aplicaciones orientadas a la inteligencia artificial y procesamiento masivo de información. Cuenta con un administrador de paquetes denominado conda el cual se es el responsable de la instalación, ejecución y actualización de las librerías.

CAPÍTULO 4
DESARROLLO EXPERIMENTAL

En esta sección, se describe el procedimiento llevado a cabo para el desarrollo de la aplicación móvil. Dado que el propósito de este trabajo es el identificar defectos superficiales presentes en los procesos de fabricación de materias primas que involucran el uso de lámina acero o aluminio, se realizaron pruebas en tres procesos de manufactura distintos para validar las capacidades de detección de la aplicación en cada uno de estos.

Con el fin de distinguir cada proceso, se identificaron como proceso A, B y C, los cuales, en su respectivo apartado, se dará una descripción detallada de cada uno de ellos. Fue necesario obtener un set de datos de los materiales a analizar en cada proceso, así como llevar a cabo las tareas concernientes al entrenamiento del modelo, los ajustes de la aplicación y la comprobación de funcionalidad en operación.

4.1. CREACIÓN Y GESTIÓN DEL SET DE DATOS

Una de las etapas con más importancia al implementar una red neuronal es la adquisición de los datos que serán utilizados en la fase de entrenamiento. Es necesario establecer el objetivo principal de la implementación, analizar cómo se manejan los datos durante el proceso y establecer estándares de configuración para tratar varios problemas del detector de imágenes.

4.1.1. Set de datos para el proceso A

Como primera etapa, con el apoyo de la empresa **Materiales y Molduras de Mexicali**, se estudió el proceso de manufactura de polín estructural de 3 x 4 pulgadas, fabricado con lámina de acero galvanizado calibre 24, utilizado para la elaboración de estructuras en la construcción de edificaciones.

En este proceso, se identificaron dos defectos que comúnmente se presentan durante la fabricación del material, el primero de ellos se presentó en forma de un doblez en los bordes laterales de la pieza durante el troquelado del material, el segundo defecto identificado, fue una especie de plegado del material, formando arrugas fácilmente apreciables en los costados del producto final.

Las razones por las que se puede presentar este tipo de defectos durante el proceso de fabricación son diversas, pero en su mayoría, de acuerdo a la experiencia del personal encargado de operar la maquinaria, se deben a una mala configuración del equipo al iniciar el proceso, a la falta de mantenimiento de los componentes internos de la maquinaria, así como rodillos defectuosos o en muy mal estado que deben ser reemplazadas para su correcto funcionamiento.

Una vez identificado los defectos, así como los motivos que los originan, fue necesario clasificarlos, para lo cual se definieron las clases "*bent*" y "*creased*" respectivamente, que servirán como identificadores de los defectos durante el desarrollo de la aplicación.

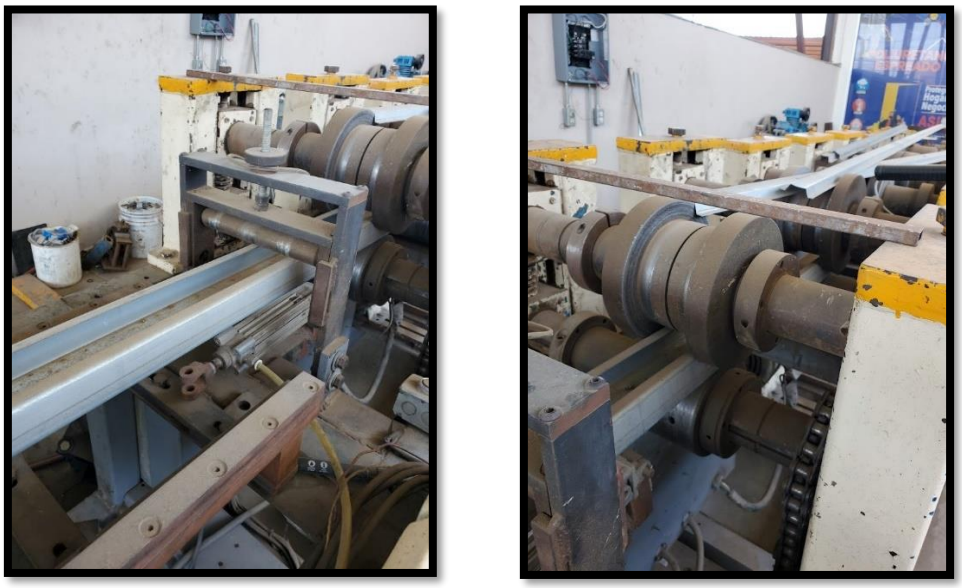

Figura 4. 1. Maquinaria para la fabricación de polín estructural con lámina de acero.

Las imágenes necesarias para el procesamiento y entrenamiento del algoritmo de detección, fueron obtenidas de material de desecho (scrap) que la empresa almacena como material de segunda línea, las capturas permitiendo identificar los defectos anteriormente mencionados. El set de datos se conformó de 180 imágenes captadas con la cámara integrada

de un equipo Samsung Galaxy Note 10+, con una relación de aspecto 1:1 y utilizando la resolución mínima permitida por el dispositivo, la cual es de 3024 x 3024 pixeles.

Dado que las imágenes son el dato de entrada a la red neuronal para su entrenamiento, deben ser redimensionadas de acuerdo a los requisitos del modelo a utilizar, por lo tanto, todas deberán tener una relación 1:1 entre sí, ya que esto traerá beneficios en la precisión del detector, por lo que los datos resultarán más eficientes.

Con el uso de la herramienta Annotate, disponible a través del portal Roboflow, se procedió a etiquetar cada una de las imágenes, con sus respectivas clases, las cuales serán utilizadas durante el entrenamiento del modelo. TensorFlow recibe la información de las etiquetas ("*bent*", "*creased*") anteriormente definidas. En la Figura 4.2 se ilustran algunas imágenes pertenecientes en el set de datos.

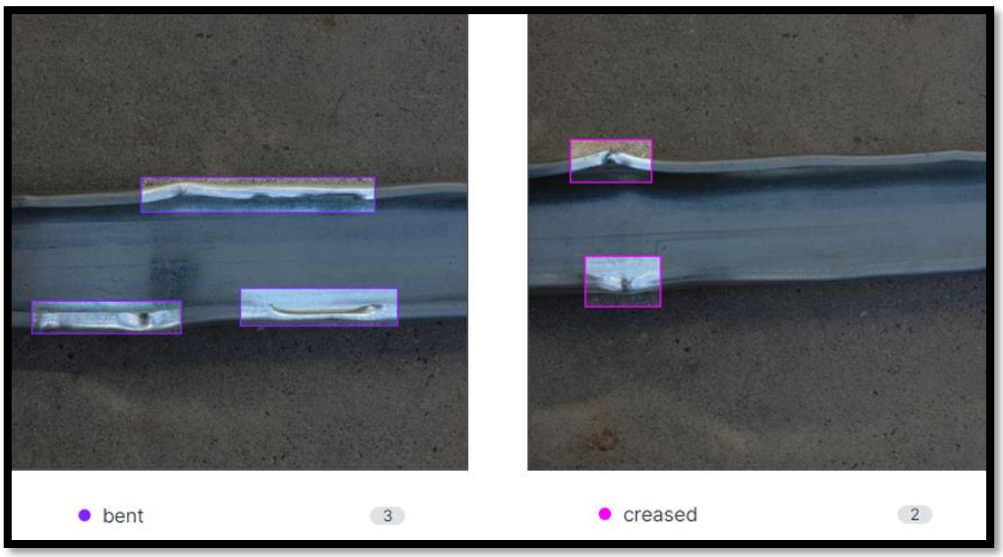

Figura 4. 2. Ejemplos de algunas de las imágenes contenidas en el set de datos para el proceso A.

En general, el set de datos se organiza en tres subconjuntos: conjunto de validación, entrenamiento y de prueba. El subconjunto de entrenamiento contiene imágenes que son utilizadas para que la red neuronal aprenda. Por otro lado, el subconjunto de validación es una agrupación de imágenes que el modelo no vio consideró en el proceso de entrenamiento,

es el encargado de efectuar la evaluación de la red y que nos permite hacer el ajuste de los parámetros correctamente. Por último, el subconjunto de prueba, que son utilizadas solamente para evaluar la eficiencia del detector y por consiguiente obtener un resultado general del mismo.

Así, de un set de datos formado que está conformado por una cantidad N de imágenes (Figura 4.3, parte superior), puede representarse la división de estos subconjuntos en tres recopilaciones de imágenes de tamaño A, B y C (Figura 4.3, parte inferior).

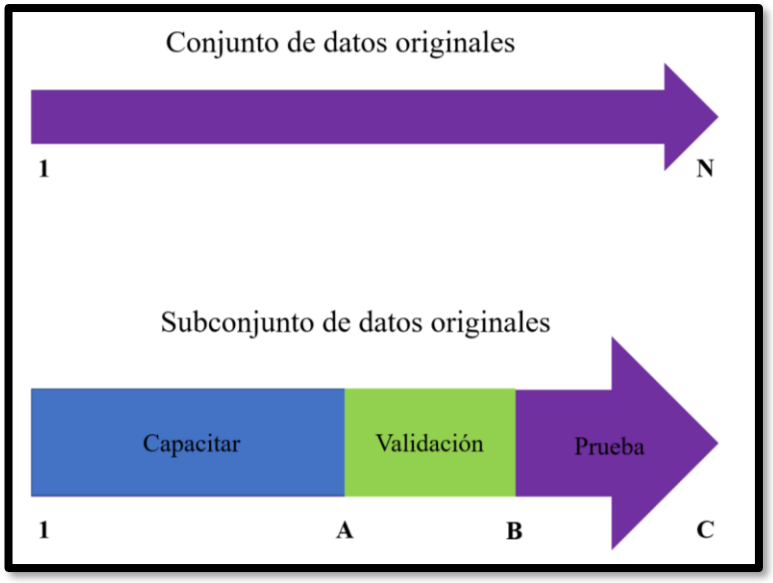

Figura 4. 3. Organización de un set de datos (arriba) en tres subconjuntos de datos (abajo) [66].

Por lo tanto, de las imágenes obtenidas, con la ayuda de la herramienta Annotate, se destinó el 70% al subconjunto *train*, 20% al subconjunto de *validation*, y finalmente 10% al subconjunto *test*. Por tanto, se obtiene una colección para entrenamiento de 126 imágenes, 36 para validar y 18 para pruebas (Figura 4.4).

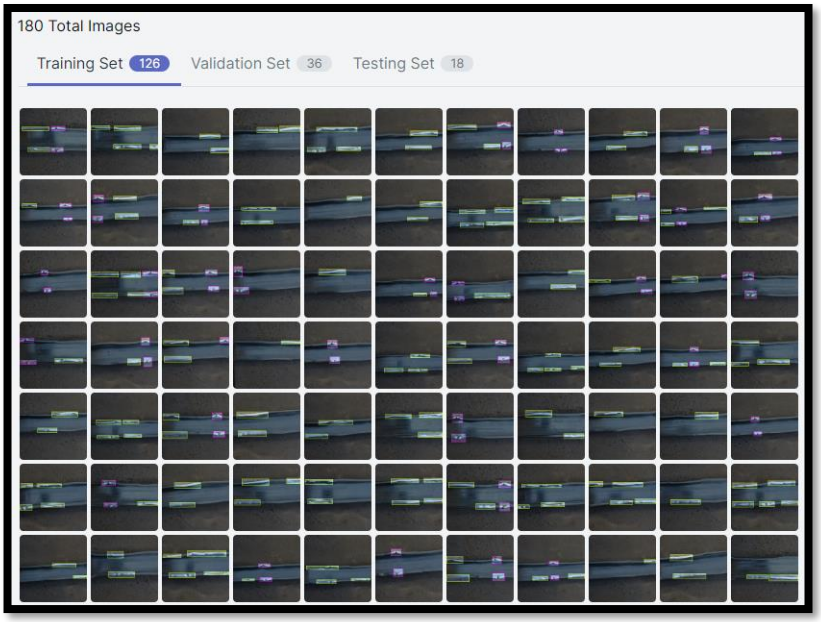

Figura 4. 4. Distribución de los subconjuntos de datos del proceso A con la herramienta Annotate.

Una vez que las imágenes fueron debidamente etiquetadas con sus correspondientes clases, y distribuidas en los subconjuntos antes mencionados, Roboflow permite exportar el set de datos en distintos formatos.

Para entrenar el modelo con Tensorflow, fue necesario trabajar con el formato Pascal VOC, por sus siglas en inglés (*Visual Object Classes*), formato estandarizado para el reconocimiento de clases de objetos. Las etiquetas están contenidas en formato XML, por sus siglas en inglés (Extensible Markup Language), que mantienen información relacionada a la clase, el cuadro delimitador y nombre de la imagen.

Además de seleccionar el formato del set de datos, es necesario redimensionar las imágenes a manera que puedan ser utilizadas por el modelo a entrenar, Roboflow permite redimensionar el tamaño de la imagen a la salida al momento de realizar la exportación del set de datos, por lo que siguiendo las especificaciones del modelo utilizado para entrenar la red, se define un tamaño de imagen de salida de 448 x 448 pixeles, resolución recomendada en la documentación oficial de Tensorflow para obtener resultados óptimos.

Finalmente, Roboflow permite descargar el set de datos en un solo archivo comprimido, con las imágenes y sus respectivos archivos de anotaciones XML, distribuidos en tres subcarpetas, *train*, *test* y *valid*, estructura de archivos requerida en la etapa de entrenamiento del modelo.

4.1.2. Set de datos para el proceso B

El segundo proceso estudiado fue en la empresa **Precisión Sheet Metal** de México, correspondiente a la fabricación de gabinetes elaborados con lamina de acero inoxidable, utilizados para el montaje de quipo electrónico en la industria, primordialmente en el sector alimentario, industria del transporte y telecomunicaciones.

En el proceso de fabricación de estos gabinetes, se utilizan 8 plantillas (48112, 48113, 48114, 48115, 48116, 48117, 48118, 48119), cada plantilla corresponde a un modelo particular de gabinete, por lo que cada gabinete, al final de la línea de producción, tendrá una configuración única que lo caracteriza entre los demás modelos. Las características que los diferencian son mínimas, como la posición de algunos remaches en distintos puntos del gabinete según el modelo, la ubicación de diversos orificios que sirven de ventilación o que permiten la introducción de cables en ellos, así como la forma de la base central del gabinete.

Dado que, a simple vista, para un operador poco experimentado, es difícil diferenciar entre cada tipo de gabinete, es necesario probar en cada uno de ellos las 8 plantillas disponibles con el fin de identificar a qué modelo pertenece, ya que comparten características y dimensiones muy similares, motivo por el cual la empresa tiene la necesidad de lograr identificar de manera automatizada cada gabinete que sale de la línea de producción y que es puesto bajo la revisión del inspector de calidad. Además de garantizar que el acabado superficial final de cada gabinete, se encuentre dentro de los requisitos que el cliente solicita, por lo que el producto final debe estar libre de marcas visibles causadas durante la última etapa del proceso, la cual consiste en dar acabado mediante una lijadora industrial operada por un técnico, siendo este último, el responsable de lograr el acabado final.

Figura 4. 5. Proceso de lijado para la obtención del acabado final del gabinete.

Para realizar la captura de imágenes se tomaron como muestra 2 modelos de gabinete (48117 y 48118) de los 8 modelos que la empresa fabrica, esto debido a que, al momento de la investigación, la empresa solo disponía de estos 2 modelos en su inventario. De cada modelo, se identificaron los rasgos más significativos, clasificados como *"48117_a"*, que identifica el tipo de soporte central en el gabinete, *"48117_b"* para identificar la posición de los remaches a un costado del soporte central y *"48117_c"* que identifica la posición de los orificios en la parte superior del gabinete. Estos tres rasgos son fundamentales para diferencial entre ambos modelos, por tal motivo, también fueron identificados en el gabinete 48118, con sus respectivos identificadores *"48118_a", "48118_b" y "48118_c"*.

Figura 4. 6. Gabinete modelo 48117 (izquierda) y gabinete modelo 48118 (derecha).

Estos identificadores describen las características más significativas que diferencian un modelo de otro, ya que a simple vista tienen gran similitud. Además, se tomaron capturas de la parte posterior del gabinete, con el fin de evidenciar las marcas que no deberían estar presentes una vez realizado el acabado final, y de esta forma, la aplicación sea capaz de identificar este tipo de marcas en el producto final.

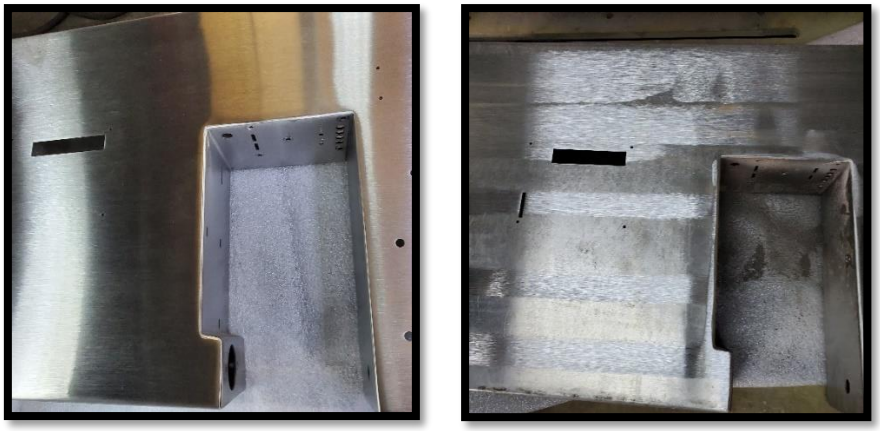

Figura 4. 7. Acabado final esperado del producto (izquierda), marcas causadas por la lijadora (derecha).

Al igual que en el proceso descrito en el apartado anterior, algunas de las imágenes fueron obtenidas de material de que no cumplió con los requisitos de calidad y otras más, fueron tomadas de gabinetes de primera línea para la identificación de los distintitos rasgos para cada modelo.

El set de datos se integró de 544 imágenes captadas con la cámara integrada de mismo equipo Samsung Galaxy Note 10+ utilizado en el proceso A, respetando la misma relación de aspecto 1:1 y utilizando la resolución mínima permitida por el dispositivo, la cual es de 3024 x 3024 pixeles.

Con el uso de la herramienta Annotate disponible a través del portal Roboflow, se procedió a etiquetar cada una de las imágenes, con sus respectivas clases, las cuales son utilizadas durante la etapa de entrenamiento del modelo. En la figura 4.8 se observan algunos ejemplos de imágenes contenidas en el set de datos.

Figura 4. 8. Ejemplos de algunas de las imágenes contenidas en el set de datos para el proceso B.

Con la ayuda de la herramienta Annotate, se destinó el 80% al subconjunto *train*, 10% al subconjunto *validation* y finalmente 10% al subconjunto *test*. Por tanto, se obtuvo una colección de 432 imágenes para entrenar, 56 imágenes para validar y 56 imágenes para pruebas (figura 4.9).

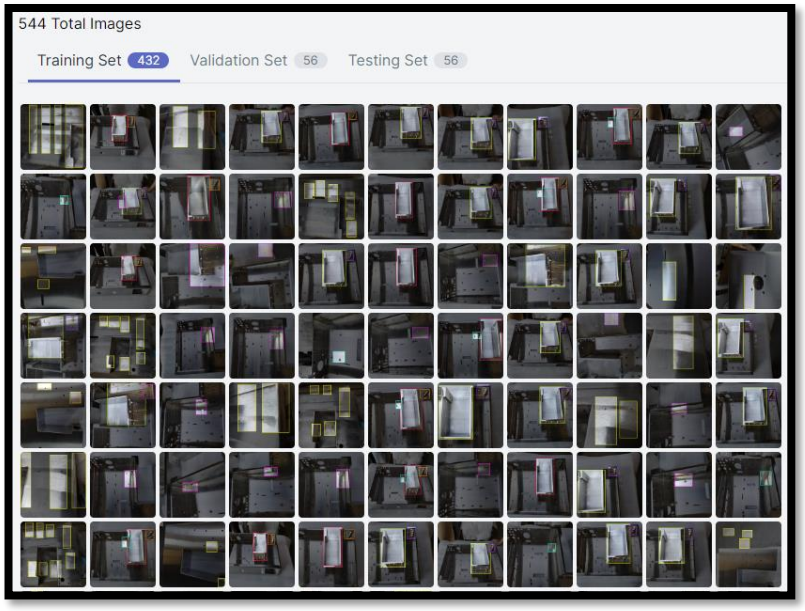

Figura 4. 9. Distribución de los subconjuntos de datos del proceso B con la herramienta Annotate.

Una vez finalizada la tarea de etiquetado de las imágenes con sus respectivas clases, y distribuidas en los subconjuntos antes mencionados, Roboflow permite exportar el set de datos en distintos formatos, para entrenar el modelo con Tensorflow, fue necesario trabajar con el formato Pascal VOC anteriormente utilizado. Finalmente, se exporta el set de datos en un solo archivo comprimido, con las imágenes y sus respectivos archivos de anotaciones XML, distribuidos en tres subcarpetas, *train*, *test* y *valid*, estructura de archivos requerida en la etapa de entrenamiento del modelo.

4.1.3. Set de datos para el proceso C

El último proceso estudiado, fue posible con el apoyo de la empresa **Esliter de México**, el cual consiste en la fabricación de tubo redondo industrial de acero de 1-5/8 pulgadas de diámetro, elaborado con lamina calibre 18, utilizado en la elaboración de estructuras para diversas aplicaciones. En este proceso, se identificaron tres defectos comunes que se hacen ver durante la manufactura del material, por lo que se definieron las clases a etiquetar: *"weld_defect"*, *"weld_joint"* y *"hole"*.

Estos identificadores describen los defectos más comunes en el proceso de fabricación. La clase *"weld_defect"* obedece a discontinuidades en la aplicación de la soldadura al unir el material y formar el tubo redondo, el segundo identificador *"weld_joint"*, se refiere a la presencia de uniones entre dos segmentos de material, este no es considerado un defecto como tal, ya que la unión se realiza de manera intencional por parte del operador, con el fin unir el segmento final de una bobina de lámina que esta por agotarse, con el segmento de una nueva bobina, esto con el fin de tener continuidad en el proceso y evitar desperdicios de material. El tercer identificador *"hole"* se refiere a defectos causados principalmente por una mala configuración del equipo, al desgaste de los rodillos de la maquinaria o al mal funcionamiento del electrodo al momento de realizar la soldadura.

Figura 4. 10. Maquinaria para la fabricación de tubo redondo industrial con lámina de acero.

Al igual que en los procesos descritos en los apartados anteriores, las imágenes fueron tomadas de material de desecho (scrap), lo que permitió destacar en cada una de ellas los defectos anteriormente mencionados.

El set de datos está compuesto de 430 imágenes captadas con la cámara integrada de mismo equipo Samsung Galaxy Note 10+ utilizado en los procesos anteriores, con la misma relación de aspecto 1:1 y utilizando la resolución mínima permitida por el dispositivo de 3024 x 3024 pixeles.

Con el uso de la herramienta Annotate, se procedió a etiquetar cada una de las imágenes, con sus respectivas clases, las cuales serán utilizadas durante el entrenamiento del

modelo. TensorFlow recibe la información de las etiquetas *("weld_defect", "weld_joint"* y *"hole"*). En la figura 4.11 se muestran ejemplos de imágenes contenidas en el set de datos.

Figura 4. 11. Ejemplos de algunas de las imágenes contenidas en el set de datos para el proceso C.

Utilizando la herramienta Annotate, se destinó el 70% al subconjunto de *train*, 20% al subconjunto de *validation*, y finalmente 10% al subconjunto *test*. Por tanto, se obtuvo una colección de 300 imágenes para entrenar, 87 imágenes para validar y 43 imágenes para pruebas (figura 4.12).

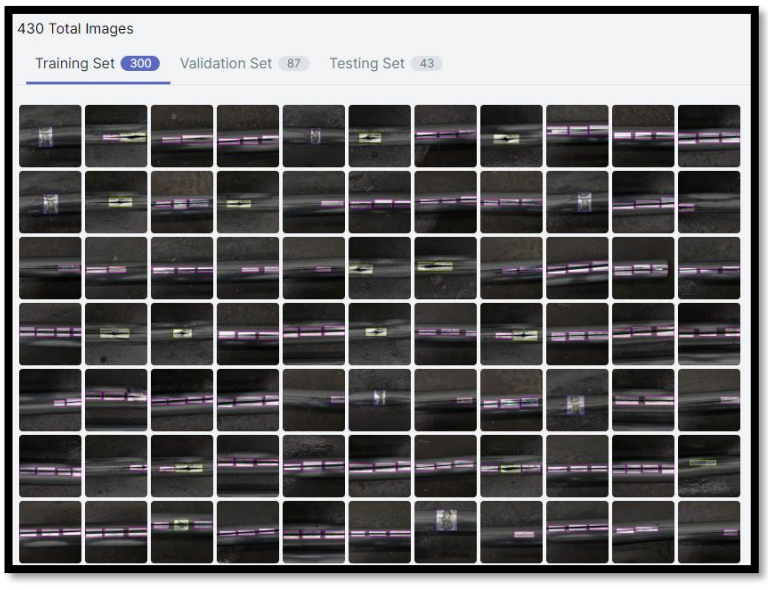

Figura 4. 12. Distribución de los subconjuntos de datos del proceso C con la herramienta Annotate.

Una vez que las imágenes han sido debidamente etiquetadas con sus correspondientes clases, y distribuidas en los subconjuntos antes mencionados, Roboflow permite exportar el set de datos en distintos formatos, para entrenar del modelo con Tensorflow, fue necesario trabajar con el formato Pascal VOC. Finalmente, se exporta el set de datos en un solo archivo comprimido, con las imágenes y sus respectivos archivos de anotaciones XML, distribuidos en tres subcarpetas, *train*, *test* y *valid*, estructura de archivos requerida en la etapa de entrenamiento del modelo.

4.2. CONFIGURACIÓN DEL FRAMEWORK

El marco elegido para implementar redes neuronales es TensorFlow, que requiere Anaconda para Python y sus bibliotecas y CUDA para optimizar las fases de procesamiento y entrenamiento de imágenes. Para entrenar el modelo de detección de objetos personalizado mediante una agrupación de imágenes de entrenamiento, TensorFlow tiene una biblioteca TFLite Model Maker que simplifica el proceso de entrenamiento de modelos de TensorFlow Lite. Utiliza el entrenamiento de transferencia para reducir la cantidad de datos de entrenamiento requeridos y acortar el tiempo de entrenamiento. Para el proceso de entrenamiento del modelo, se utilizó Google Colab, que utiliza sus computadoras portátiles para ejecutar código en servidores alojados en la nube de Google, lo que significa un uso máximo del hardware de Google, incluidas GPU y TPU, sin importar qué tan potente sea nuestro equipo, todo lo que necesita es un navegador y acceso a Internet.

4.2.2. TensorFlow en modo GPU

Es posible instalar TensorFlow en dos modos diferentes: modo CPU y modo GPU. En el modo CPU se asume una instalación básica con los requisitos básicos donde toda la ejecución se maneja en la CPU de forma directa. En el segundo modo, si el dispositivo tiene una GPU dedicada, el modo GPU puede acelerar el proceso realizado por TensorFlow. Dado que las máquinas virtuales utilizadas en Google Colab cumplen los requisitos para usar

TensorFlow en modo GPU, elegimos usar este modo. La tecnología detrás de este modelo se llama Compute Unified Device Architecture (CUDA), la cual hace alusión a una plataforma de cómputo paralela basada en la codificación algorítmica de las GPU NVIDIA del fabricante. Para realizar una instalación completa en modo GPU de TensorFlow se requiere el kit de herramientas CUDA y las bibliotecas CuDNN. Finalmente, se siguen los pasos del framework en su documentación oficial para instalarlo.

4.2.3. TensorFlow Lite Model Maker

Es una biblioteca de Python que simplifica significativamente la etapa del entrenamiento del modelo de aprendizaje automático mediante un conjunto de datos personalizado. Aprovecha el aprendizaje por transferencia para permitir el entrenamiento de modelos de alta calidad utilizando solo un conjunto de imágenes y acortar el tiempo de entrenamiento[67].

Hay dos formas de instalar Model Maker.

1. Instalar un paquete pip pre-compilado.

```
pip install tflite-model-maker
```

Si se desea instalar la versión nocturna, se utiliza el comando:

```
pip install tflite-model-maker-nightly
```

2. Clonar el código fuente de GitHub e instalarlo.

```
git clone https://github.com/tensorflow/examples
cd examples/tensorflow_examples/lite/model_maker/pip_package
pip install -e
```

TensorFlow Lite Model Maker depende del paquete pip de TensorFlow para su instalación. Para los controladores de GPU, es necesario consultar la guía de GPU o la guía de instalación de TensorFlow.

4.3. COMPILACIÓN E IMPLEMENTACIÓN DE UN MODELO DE DETECCIÓN DE OBJETOS PERSONALIZADO POR *TRANSFER LEARNING*

La técnica utilizada para hacer la implementación de este sistema se denomina transferencia de aprendizaje. El método usa modelos pre-entrenados con diferentes bases de datos y se enfoca en modificar la última capa de la red neuronal. Como se mencionó anteriormente, estas capas son parte de un conjunto más especializado de funciones de CNN. Con el apoyo de esta técnica es posible enfocarse en los hiperparámetros más importantes del conjunto de datos, que se pueden crear a partir del conjunto de datos más pequeño. También reduce significativamente los cálculos computacionales debido a que el modelo utilizado está pre-entrenado. Por lo tanto, este método también se conoce como rehabilitación modelo. Para la implementación del respectivo modelo se siguen tres fases: fase de entrenamiento, optimización y prueba.

El primer paso es identificar el modelo inicial que se utilizará, los hiperparámetros explorados para refinar la red y la transformación de datos adecuada. La fase de optimización implica refinar la selección de parámetros determinada en el paso anterior para mejorar el resultado y alcanzar una precisión altamente confiable. La etapa final consta de realizar algunas pruebas para validar el resultado obtenido durante el entrenamiento y la optimización, ya que una alta precisión no necesariamente significa que se obtenga el resultado esperado. En todo caso, sería necesario volver a la etapa de optimización para buscar obtener un mejor resultado.

4.3.1. Fase de entrenamiento

TensorFlow incluye un conjunto de herramientas diseñadas para volver a entrenar modelos mediante técnicas de aprendizaje por transferencia. En la implementación del modelo se siguió un ejemplo de detección de objetos proporcionado en la documentación oficial de Tensorflow, para lo cual se utilizó Google Colab, que permite escribir y ejecutar código para entrenar un modelo personalizado.

Pre requisitos

Cuando tenemos acceso a una notebook virtual en Google Colab, por defecto esta ya viene con Python en su versión 3.6 previamente instalada, lo que nos ahorra el tener que llevar a cabo nosotros mismo la instalación de esta distribución, sin embargo, es necesario instalar algunos paquetes e importar las librerías necesarias para el entrenamiento del modelo. Comenzamos por instalar los paquetes necesarios, incluido el paquete Model Maker del repositorio de GitHub y la biblioteca pycocotools que usará para la evaluación.

```
pip install -q tflite-model-maker
pip install -q pycocotools
```

Posteriormente, es necesario importar los paquetes y librerías a nuestro entorno de trabajo.

```
import numpy as np
import os

from tflite_model_maker.config import ExportFormat
from tflite_model_maker import model_spec
from tflite_model_maker import object_detector
import tensorflow as tf
assert tf.__version__.startswith('2')

tf.get_logger().setLevel('ERROR')
from absl import logging
logging.set_verbosity(logging.ERROR)
```

Antes de poder comenzar a crear nuestro propio detector de objetos personalizado, se deberá preparar el conjunto de datos. Tensorflow Lite Model Maker admite dos formatos de datos: CSV y Pascal VOC. Los datos en formato CSV se pueden cargar con *object_detector.DataLoader.from_csv* y los datos en formato Pascal VOC se pueden cargar con el método *object_detector.DataLoader.from_pascal_voc*, es nuestro caso, como ya se

comentó en la sección anterior, los datos fueron preparados en un formato Pascal VOC para un mejor manejo.

Pasos para el entrenamiento del modelo de detección de objetos personalizados.

1. Elección de una arquitectura de modelo de detección de objetos.

Tensorflow Lite Model Maker actualmente admite 5 modelos diferentes de detección de objetos (EfficientDet-Lite [0-4]). Todos ellos se derivan de la arquitectura EfficientDet. Las principales diferencias entre los modelos son su tamaño y latencia.

Tabla 4- 1. Rendimiento de cada modelo EfficientDet-Lite en comparación con los demás [68].

ARQUITECTURA DEL MODELO	TAMAÑO(MB)*	LATENCIA(MS)**	PRECISIÓN MEDIA***
EfficientDet-Lite0	4.4	37	25.69%
EfficientDet-Lite1	5.8	49	30.55%
EfficientDet-Lite2	7.2	69	33.97%
EfficientDet-Lite3	11.4	116	37.70%
EfficientDet-Lite4	19.9	260	41.96%

* Tamaño de los modelos cuantificados enteros.

** Latencia medida en Pixel 4 usando 4 subprocesos en la CPU.

*** La precisión promedio es el mAP (precisión promedio promedio) en el conjunto de datos de validación de COCO 2017.

Como se puede ver en la Tabla 4-1, existe una familia de modelos previamente entrenados, disponibles para ser reentrenados con nuestro set de datos, los modelos están optimizados para ejecutarse en dispositivos móviles, sin embargo, varían en función de su

tamaño en megabytes, latencia y precisión media, entre más ligero es el modelo, menor será su latencia, ya que requiere menos operaciones computacionales, como es el caso del primer modelo disponible EfficientDet-Lite0, sim embargo, su ligereza repercute en su precisión al momento de detectar objetos sobre las imágenes o el flujo de video, por otra parte, tenemos el modelo más robusto EfficientDet-Lite4, el cual, en términos de tamaño, es entre 4 y 5 veces más pesado que EfficientDet-Lite0, razón por la cual la latencia se incrementa, ya que el modelo requiere de más recursos computacionales para ser procesado, sin embargo, su capacidad de identificar objetos es superior a todos los demás modelos.

Los modelos más ligeros, son optimizados para equipos móviles con prestaciones limitadas de hardware, mientras que los modelos más robustos, están enfocados a equipos móviles con altas prestaciones de hardware, con capacidades de procesamiento elevadas y con acceso a procesamiento apoyado en GPU.

El dispositivo a utilizar para ejecutar la aplicación móvil es un Samsung Galaxy Note 10+, con un procesador Cortex de 8 núcleos, 12 GB de memoria RAM y soporte para procesamiento con GPU, capacidades técnicas por encima de lo recomendado, de acuerdo a los resultados de las pruebas realizadas a los modelos, motivo por el cual se decidió utilizar el modelo EfficientDet-Lite3 por su balance entre latencia y precisión.

Para la selección del modelo se ejecuta la siguiente línea de comando.

```
spec = model_spec.get('efficientdet_lite3')
```

2. Cargar el conjunto de datos.

Model Maker tomará los datos de entrada en formato Pascal VOC. Se utiliza el método *object_detector.DataLoader. from_pascal_voc* para cargar el conjunto de datos y dividirlos en imágenes de entrenamiento, validación y prueba.

- Imágenes de entrenamiento: estas imágenes se utilizan para efectuar el entrenamiento del modelo de detección de objetos para que reconozca los rasgos de interés en las imágenes.

- Imágenes de validación: estas son imágenes que el modelo no vio durante el proceso de entrenamiento. Los usará para decidir cuándo debe detener el entrenamiento, para evitar el sobreajuste.
- Imágenes de prueba: estas imágenes se utilizan para evaluar el desempeño final del modelo.

Se puede cargar el archivo data.zip directamente desde Google Cloud Storage, no es necesario mantener las imágenes en Google Cloud para usar Model Maker, es posible especificar un archivo local en el equipo y Model Maker funcionará bien.

Para cargar el archivo se ejecuta la siguiente instrucción.

```
train_data = object_detector.DataLoader.from_pascal_voc('/content/train', '/content/train', label_map={1: 'bent',2: 'creased'} )
```

3. Entrena el modelo de TensorFlow con los datos de entrenamiento.

Después de cargar los datos, el modelo de Tensorflow se puede entrenar con el método *object_detector.create*. El método *create* es la función que utiliza la biblioteca Model Maker para crear modelos. El método de *create*:

- Crea el modelo para la detección de objetos según *model_spec*.
- Entrena al modelo. De forma predeterminada, se utilizan los hiperparámetros dentro de *model_spec*, pero se pueden sobrescribir pasando los hiperparámetros como argumentos de función.

Entre los hiperparámetros más significativos del modelo EfficientDet-Lite3 encontramos *epochs = 50* de forma predeterminada, lo que significa que, durante el entrenamiento, se pasará por el conjunto de datos 50 veces. Se puede observar la precisión de la validación durante el entrenamiento y detenerse antes para evitar el sobreajuste. Se puede mejorar la precisión del detector aumentando el número de épocas, pero de igual manera, se puede dar lugar a un sobreajuste.

Otro parámetro de importancia es *batch_size = 64* de forma predeterminada, este parámetro indica el conjunto de imágenes que son utilizadas en un solo paso de entrenamiento, se necesitan 3 pasos para recorrer 180 imágenes en el set de datos para entrenamiento. Se puede reducir este parámetro con el fin de disminuir la cantidad de memoria utilizada por el equipo de cómputo utilizado. Finalmente, *train_whole_model = False* por defecto, habilitando este parámetro, nos permite entrenar todo el modelo en lugar de solo entrenar la capa principal, esto con el fin de mejorar la precisión. La contrapartida es que puede llevar más tiempo entrenar el modelo.

Siguiendo las recomendaciones de la documentación oficial de Tensorflow, se invocó la función *create* con los parámetros por defecto.

```
model = object_detector.create(train_data, model_spec=spec, validation_data=validation_data)
```

Obteniendo la siguiente salida:

```
Epoch 1/50
21/21 [==============================] - 79s 2s/step - det_loss: 1.7540 - cls_loss: 1.1316 - box_loss: 0.0124 - reg_l2_loss: 0.0764 - loss: 1.8305 - learning_rate: 0.0090 - gradient_norm: 0.7342 - val_det_loss: 1.6683 - val_cls_loss: 1.1064 - val_box_loss: 0.0112 - val_reg_l2_loss: 0.0764 - val_loss: 1.7447
Epoch 2/50
21/21 [==============================] - 28s 1s/step - det_loss: 1.5948 - cls_loss: 1.0771 - box_loss: 0.0104 - reg_l2_loss: 0.0764 - loss: 1.6713 - learning_rate: 0.0100 - gradient_norm: 1.0794 - val_det_loss: 1.4721 - val_cls_loss: 0.9567 - val_box_loss: 0.0103 - val_reg_l2_loss: 0.0764 - val_loss: 1.5485
Epoch 3/50
21/21 [==============================] - 32s 2s/step - det_loss: 1.3778 - cls_loss: 0.9399 - box_loss: 0.0088 - reg_l2_loss: 0.0764 - loss: 1.4542 - learning_rate: 0.0099 - gradient_norm: 1.9733 - val_det_loss: 1.2991 - val_cls_loss: 0.8070 - val_box_loss: 0.0098 - val_reg_l2_loss: 0.0764 - val_loss: 1.3755
```

En la Tabla 4-2, se enlistan los argumentos que se pueden configurar correspondientes al método *create* con su respectiva descripción.

Tabla 4- 2. Argumentos del método "tflite_model_maker.object_detector.create"[69].

ARGUMENTO	DESCRIPCIÓN
-- *train_data*	Datos de entrenamiento.
-- *model_spec*	Especificación para el modelo.

-- *validation_data*	Datos de validación. Si es nulo, salta el proceso de validación.
-- *epochs*	Número de épocas para el entrenamiento.
-- *batch_size*	Tamaño del lote para entrenamiento.
-- *train_whole_model*	Falso por defecto. Si es verdadero, entrenar todo el modelo.
-- *do_train*	Si es verdadero, ejecutar el entrenamiento.

Existen múltiples configuraciones para los modelos EfficientDet-Lite. Por una parte, se puede elegir la resolución de la imagen de entrada con valores de 320, 384, 448, 512 o 640 píxeles. Estos valores son determinísticos para el tiempo de procesamiento y la precisión de la red, cuanto mayor sea la resolución, mayor será el tiempo de entrenamiento y mayor será la precisión. Por otro lado, puede elegir el tamaño relativo del modelo como se describió anteriormente en la arquitectura del modelo.

Al finalizar el proceso de transferencia de aprendizaje, se genera un nuevo modelo en relación a las etiquetas definidas en la ubicación correspondiente y se exporta el modelo de directorio *"export_dir"*. Ya que se utilizaron los parámetros que la función tiene por defecto, es posible que el resultado obtenido no sea los más óptimo, como se muestra a continuación.

```
{'AP': 0.22944169,
'AP50': 0.7245343,
'AP75': 0.07708805,
'AP_/bent': 0.17161024,
'AP_/creased': 0.28727314,
'APl': -1.0,
'APm': 0.23059428,
'APs': -1.0,
'ARl': -1.0,
'ARm': 0.41484848,
'ARmax1': 0.1509091,
'ARmax10': 0.38454545,
'ARmax100': 0.41484848,
'ARs': -1.0}
```

Con la intención de tener una mejora en el resultado, se realizó una fase de optimización en búsqueda de una variación de parámetros.

4.3.2. Fase de optimización

Durante esta etapa se propuso variar tres parámetros: *"epochs"*, *"batch_size"* y *"train_whole_model"*. Como se mencionó anteriormente, *"epochs"* permite definir la cantidad de veces que, durante el entrenamiento, se pasará por el set de datos, el valor por defecto para este parámetro es de 50 épocas, de acuerdo a la documentación oficial de Tensorflow, sin embargo, se recomienda incrementar este parámetro con el fin permitir que la red mejore su capacidad de detección, esto conlleva a un mayor tiempo de procesamiento, ya que por cada época, se analiza en su totalidad el set de datos de entrenamiento.

Según el autor Rajdeep Dua en su libro Mastering TensorFlow 2.x: Implement Powerful Neural Nets[70], comenta que la precisión deja de mejorar significativamente después de 80 épocas. Después de eso, comienza a sobre ajustarse. Basándose en los hallazgos del autor, se decidió aumentar a 80 épocas el entrenamiento, con el fin de incrementar la capacidad de detección de la red.

Con lo que respecta a *"batch_size"*, se conoce como el conjunto de muestras que es utilizado durante las iteraciones al momento del entrenamiento del modelo (su valor puede comprender desde el 1 y 1000), cuanto mayor sea el tamaño del lote, más espacio de memoria necesitará, su valor por defecto es de 64.

Al disminuir este parámetro, se requerirá menos memoria, dado que entrena la red con menos muestras, el procedimiento de entrenamiento general requiere menos memoria, eso es especialmente importante si no se puede colocar todo el set de datos en la memoria del equipo. Por lo general, las redes se entrenan más rápido con mini lotes. Eso es porque se actualizan los pesos después de cada propagación. En este caso, para un set de datos de 180 imágenes, se propagan 3 lotes (2 de ellos con 64 muestras y 1 con 52 muestras) y después de cada uno de ellos se actualizan los parámetros de la red. Si usáramos todas las muestras

durante la propagación, haríamos solo 1 actualización para el parámetro de la red. Cada que un lote se procesa, se considera como un paso o step.

Con el fin de optimizar el entrenamiento y tomando en cuenta la cantidad de muestras del primer set de imágenes para entrenamiento que consta de solo 180 muestras, se reduce este parámetro a 20, esto con el fin de hacer coincidir el número de muestras por lote con la totalidad de muestras del set de datos, requiriendo 9 pasos para procesar la totalidad de las muestras, en lugar de solo 3 pasos utilizando el valor por defecto de 64, de esta manera se incrementa el número de actualizaciones de parámetros de la red.

Finalmente, se ajusta el parámetro *train_whole_model = true*, con el fin de habilitar esta opción, que por defecto se encuentra deshabilitada. Esto permitirá entrenar todo el modelo en lugar de solo entrenar la capa principal, lo que mejora la precisión, en contra parte, se requiere más tiempo entrenar el modelo.

Con dichos hiperparámetros definidos, se ejecuta de nueva cuenta el entrenamiento de la red. El fragmento de código presentado a continuación muestra la funcionalidad descrita anteriormente.

```
model = object_detector.create(train_data, model_spec=spec, batch_size=20, train_whole_model=True,
epochs=80, validation_data=validation_data)
```

Una vez que se ejecuta el script, se lleva a cabo el entrenamiento de la red con los nuevos parámetros configurados, al fin de optimizar la red, una vez finalizado el entrenamiento se obtienen los siguientes resultados en la etapa de evaluación.

```
{'AP': 0.26214552,
 'AP50': 0.65030724,
 'AP75': 0.15742834,
 'AP_/bent': 0.1889109,
 'AP_/creased': 0.33538017,
 'APl': 0.35593718,
 'APm': 0.25478762,
 'APs': -1.0,
 'ARl': 0.53055555,
 'ARm': 0.47333333,
```

'ARmax1': 0.19818182,
'ARmax10': 0.44939393,
'ARmax100': 0.48727274,
'ARs': -1.0}

Como se puede observar en los resultados, se presentó una mejora en las capacidades de detección del modelo en la fase de optimización con la variación de los parámetros, por una parte, la capacidad media de detección para la etiqueta *bent*, presentó una mejora aproximada del 10%, respecto a los resultados obtenidos del entrenamiento previo, utilizando los valores por defecto del modelo.

Por otra parte, la capacidad de detección media para la etiqueta *creased*, también se vio beneficiada, ya que, con el ajuste de los parámetros de optimización, la capacidad de detección para esta categoría presento un incremento aproximado del 15% respecto al entrenamiento previo.

4.3.3. Fase de prueba.

La etapa de prueba es el último paso del entrenamiento. Consiste en exportar el modelo de detección de objetos entrenado al formato TensorFlow Lite, especificando a qué carpeta se desea exportar el modelo cuantificado.

Varios factores pueden afectar la precisión del modelo al exportar a TFLite:

- La cuantificación ayuda a reducir el tamaño del modelo 4 veces a expensas de una disminución de la precisión.
- El modelo TensorFlow original utiliza supresión no máxima (NMS) en cada clase para el posprocesamiento, la supresión no máxima (NMS) es una técnica utilizada principalmente en la detección de objetos que tiene por objeto seleccionar el mejor cuadro delimitador de un conjunto de cuadros superpuestos, mientras que el modelo TFLite usa NMS global, de mucha más rapidez, pero menor precisión.

Por lo tanto, se debe evaluar el modelo TFLite exportado y comparar su precisión con el modelo TensorFlow original.

Para ello se ejecuta el siguiente comando, indicando el nombre del modelo TFLIte exportado y el set de datos de prueba.

```
model.evaluate_tflite('model.tflite', test_data)
```

Obteniendo los siguientes resultados de la fase de evaluación para el modelo TFLite.

```
{'AP': 0.25358814,
 'AP50': 0.6508635,
 'AP75': 0.1303322,
 'AP_/bent': 0.18200459,
 'AP_/creased': 0.3251717,
 'APl': 0.31807923,
 'APm': 0.24526039,
 'APs': -1.0,
 'ARl': 0.44722223,
 'ARm': 0.41,
 'ARmax1': 0.20484848,
 'ARmax10': 0.4078788,
 'ARmax100': 0.42454547,
 'ARs': -1.0}
```

Los resultados de la fase de prueba del modelo TFLite que fue exportado, muestran una ligera pérdida de precisión del modelo, de aproximadamente un 2% respecto a la precisión alcanzada durante la etapa de optimización. Esta ligera pérdida de precisión es el resultado de la conversión del modelo general de Tensorflow, al modelo optimizado para dispositivos móviles Tensorflow Lite (TFLite), el cual, como se mencionó anteriormente, el modelo exportado es hasta 4 veces menor que el modelo general, a expensas de una disminución en su precisión como se observó en los resultados arrojados en las pruebas.

Finalmente, es posible descargar el archivo del modelo TensorFlow Lite usando la barra lateral izquierda de Colab. Haciendo clic con el botón derecho sobre el archivo model.tflite y eligiendo la opción de Descargar para descargarlo en la computadora local. En el siguiente paso es utilizar la API ObjectDetector de la biblioteca de tareas de TensorFlow Lite para integrar el modelo en la aplicación de Android.

4.4 DESARROLLO DE APLICACIÓN MÓVIL ANDROID

La aplicación final es desarrolló en Android, por lo que fue necesario instalar Android Studio. El equipo en el que se realizó el proyecto fue en un entorno Windows 10 Home. Además de contar con un procesador Intel® Core™ i5-7300HQ y una GPU AMD Radeon™ RX 550.

4.4.1. Descripción de la implementación práctica en móviles

El propósito es utilizar la herramienta base Tensorflow Lite para desarrollar aplicaciones móviles Android, ya que se diseñó para facilitar la integración de modelos de aprendizaje en dispositivos móviles. En esta investigación se evaluaron varios modelos previamente entrenados, la optimización es un paso adicional para mejorar el rendimiento y luego se convertirá de TensorFlow (para detección de objetos) al formato TensorFlow usando el convertidor TensorFlow Lite. Los modelos en este formato permiten el uso correctamente configurado del interpretador TensorFlow Lite, que establece modelos altamente eficientes en tipos diferentes de hardware, como dispositivos Android, siendo el caso de este desarrollo. Finalmente, el modelo se cargará desde la aplicación del dispositivo móvil y la estrategia se muestra en la Figura 4.13. en la imagen.

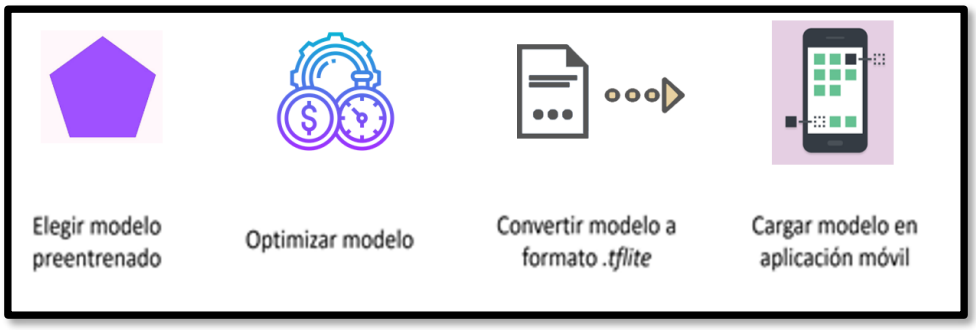

Figura 4. 13. Estrategia de implementación [71].

4.4.2. Funcionamiento de la aplicación

Ya que se crea y ejecuta la aplicación es posible crear un archivo con extensión *.apk* que sirve como instalador en el dispositivo móvil. El funcionamiento de la aplicación de detección será el siguiente:

Lo que está sucediendo en la aplicación (Figura 4.14) es la captura de recuadros (*frames*) en tiempo de ejecución utilizando la cámara integrada del dispositivo móvil, los cuales, se están convirtiendo en imágenes, para posteriormente, usarse como entrada al modelo de detección, que, a su vez, realizara las tareas de identificación de patrones en las imágenes, para finalmente, generar los valores de salida.

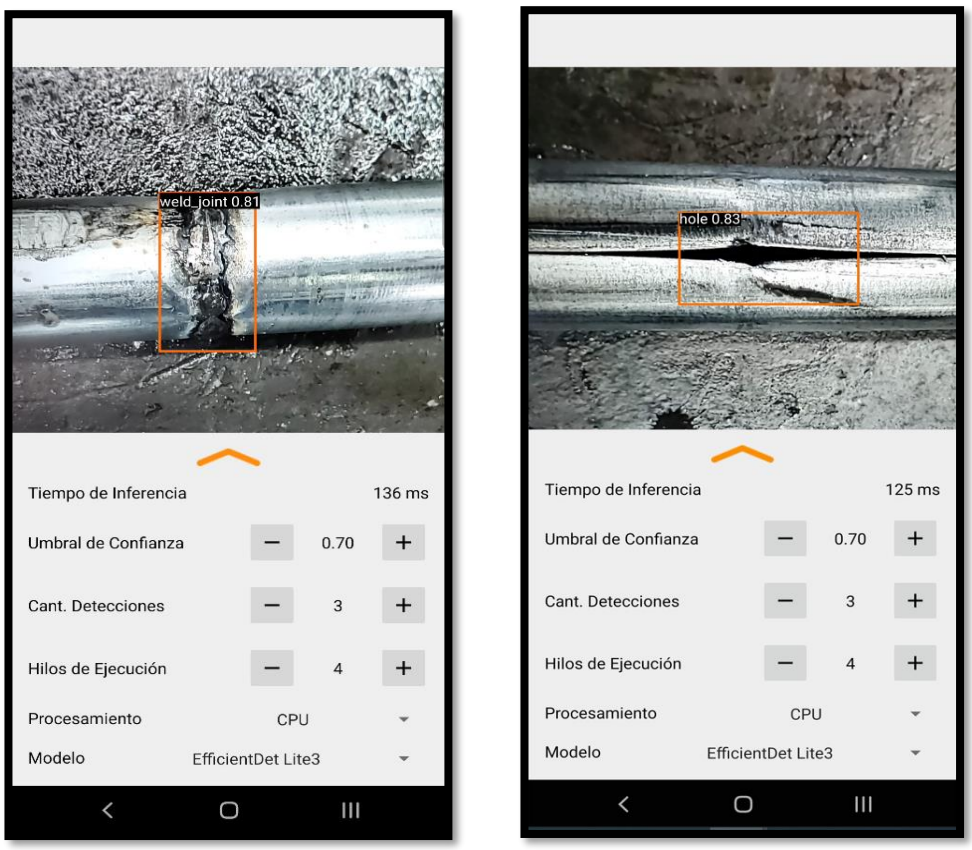

Figura 4. 14. Aplicación de detección ejecutándose en el dispositivo móvil.

Las etiquetas que se tienen como resultado son indexadas apropiadamente en conjunto con el valor de inferencia para dicha etiqueta o, dicho de otra forma, la probabilidad que el rasgo encontrado en la imagen coincida con el defecto para el cual fue entrenado, incluyendo los cuadros delimitadores para la localización de la posición de cada defecto en la interfaz, además de añadir el tiempo que le toma a la aplicación realizar la inferencia.

Esta información se muestra en la parte inferior de la Figura 4.14, donde el tiempo de inferencia se muestra primero de forma secuencial, y también se agregan cuadros delimitadores para que se pueda determinar la ubicación de cada defecto. La clase más probable de cada detección se muestra junto a los cuadros delimitadores de estos defectos detectados, lo que aumenta la probabilidad de éxito. En la sección de abajo nos encontramos con los ajustes, donde es posible establecer el umbral de confianza que va desde un valor mínimo de1 10% hasta un máximo del 80%, lo que permitirá afinar la aplicación con el fin de mostrar solo aquellos resultados que están por encima del umbral seleccionado.

El número de hilos y el tiempo de inferencia también se muestra en esta sección, que dependerá directamente de las capacidades técnicas de cada dispositivo móvil. En este caso, basándonos en las especificaciones técnicas del dispositivo móvil que se utilizaría durante las pruebas, en todo momento el número de hilos se ajustó a 4, ya que fue la configuración que mejor se adaptó al dispositivo y logro tiempos de inferencia menos prolongados.

4.4.3 Arquitectura de la aplicación

Lo siguiente es determinar el diagrama de arquitectura de la aplicación para la detección del objeto. Mirando el diagrama (Figura 4.15), podemos ver que podemos usar cualquier archivo de modelo de transformación de formato tflite y poder utilizarlo en la aplicación. Para lograrlo, se deben considerar algunos aspectos:

- Java API: contiene las librerías de clase Java para Android.
- C++ API: encargado de la carga del modelo y las invocaciones del intérprete.
- Intérprete: Responsable de la ejecución del modelo.

- API de red neuronal de Android: los dispositivos con aceleradores de hardware integrados pueden usar esta API. Esta API está actualmente disponible para modelos que ejecutan Android OS 8.1 o posterior.

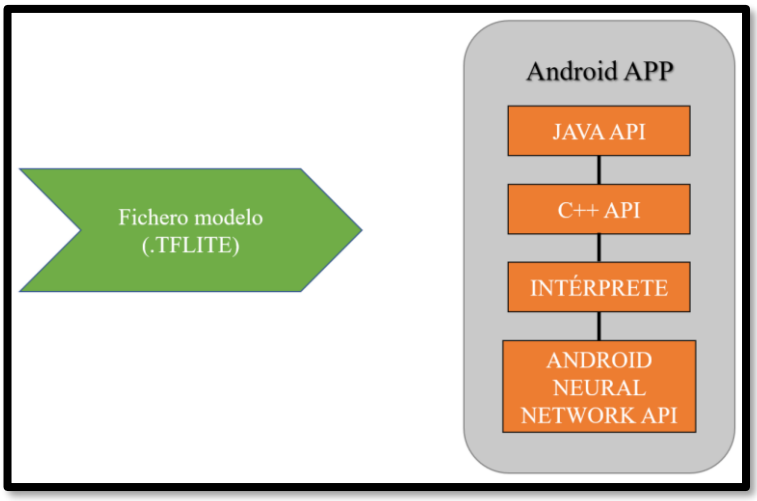

Figura 4. 15. Arquitectura general de la aplicación [71].

4.4.4. Uso de la API de TensorFlow Lite para Java

La página oficial de Tensorflow Lite proporciona una aplicación de ejemplo básica, así como documentación de la biblioteca utilizada para definir objetos, lo cual es muy útil como punto de partida para el desarrollo de aplicaciones. Actualmente, Tensorflow tiene la limitación de que solo los modelos SSD son compatibles con las tareas de detección de objetos que se ejecutan en dispositivos móviles que usan TensorFlow Lite, sin embargo, esto no presenta un inconveniente para este desarrollo, ya que el modelo utilizado está fundamentado en este principio de operación.

Dentro de la API de Tensorflow se encuentran varios ficheros utilizados en la aplicación final. En el siguiente diagrama (Figura 4.16) se muestran las clases involucradas en la detección.

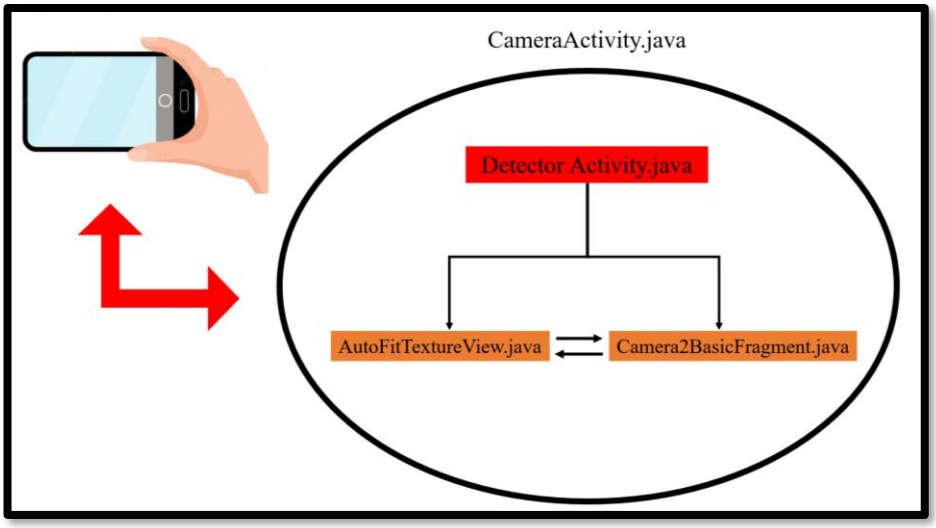

Figura 4. 16. Diagrama de clases Java de Tensorflow [71].

Normalmente, los datos de imagen recibidos de la cámara en forma de mapa de bits se toman como entrada y se convierten en un búfer de bytes para que el modelo los lea. Una vez hecho esto, el contenido de la imagen se cargará en el traductor TFLite. Detect (DetectorActivity.java) devuelve una matriz de probabilidades que tendrá cada etiqueta, que se enviará a la aplicación, incluyendo la clase con más probabilidad, el tiempo de inferencia y un objeto de cuadro delimitador que contiene la ubicación.

- AutoFitTextureView.java realiza la función de cambio de tamaño automático, que se utiliza para ajustar la relación de aspecto en el terminal, después de capturar la imagen de pantalla continua de la cámara durante la ejecución de la aplicación.
- DetectorActivity.java, su función es participar en la plataforma TensorFlow y utilizar el modelo en formato tflite. Es indispensable realizar la importación del intérprete para que cargue el modelo y efectué las operaciones en el conjunto de entrada. Después de procesar los resultados, estos se mostrarán en la parte inferior de la pantalla. La clase también necesita agregar comportamiento para manejar el uso de modelos de detección convertidos a TFLite. Esto requiere agregar los modelos mencionados y sus etiquetas correspondientes a la carpeta de archivos externos.

Además de realizar cambios en esta clase, también puede conocer la ruta al modelo (y la etiqueta correspondiente) para integrarlo en la aplicación. Si el modelo a procesar es un modelo de punto flotante o un modelo cuantificado, la diferencia es que el modelo cuantificado funcionará asignando un byte a cada valor que representa un valor entre 0 y 255. Esta diferencia debe especificarse en esta clase.

- Camera2BasicFragment.java: una clase que administra la cámara del teléfono y aprovecha el cambio de tamaño automático y la representación del resultado.
- CameraActivity.java, integra la funcionalidad de las clases anteriores (AutoFitTextureView.java, Camera2BasicFragment.java y DetectorActivity.java) y también crea una GUI que se agrega a la aplicación.

4.4.5. Interfaz gráfica de usuario

Una interfaz gráfica de usuario (GUI) interviene como mediador entre un dispositivo, llámese computadora o terminal móvil y el usuario mediante la utilización de imágenes y otros objetos. Su finalidad es mostrar de forma visual e intuitiva la información y acciones disponibles en la aplicación. En Android Studio, las GUI se pueden crear fácilmente mediante el Editor de diseño o el Editor de diseño (Figura 4.17). Las interfaces de usuario se pueden implementar gráficamente utilizando un conjunto de elementos predefinidos o codificando estos elementos en Lenguaje de marcado extensible (XML). Finalmente, se necesita hacer en la aplicación para que los usuarios puedan ver los defectos detectados en sus terminales.

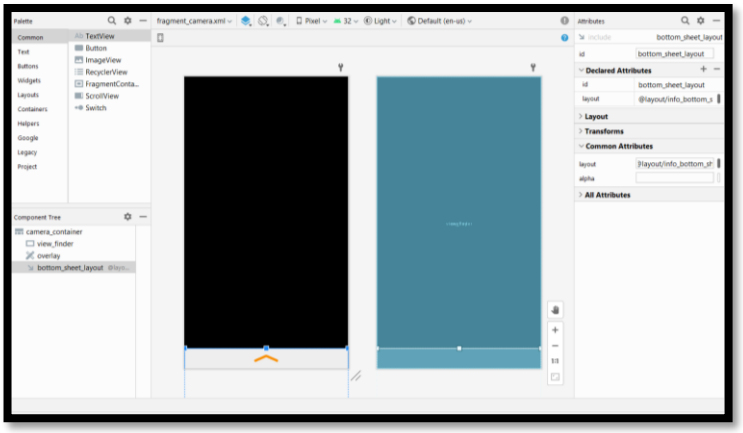

Figura 4. 17. Editor de diseños en Android Studio.

Cuando se ejecuta la aplicación, la cámara frontal del terminal se activa y aparecen los defectos detectados en la pantalla (Figura 4.18). La imagen en la pantalla corresponde a la clase "Camera2BasicFragment.java" y el texto es la cadena de etiquetas identificada por "DetectorActivity.java". El nombre elegido para la aplicación es "Sistema de detección" y una vez instalada en el dispositivo, se puede acceder a ella desde el menú de la aplicación del dispositivo.

Figura 4. 18. Aplicación de detección de defectos en Android.

CAPÍTULO 5
RESULTADOS

En esta sección se presentan los resultados obtenidos de las pruebas de campo, donde se evaluaron las capacidades del modelo en cada uno de los procesos para detectar los defectos y rasgos particulares para los que fue previamente entrenado.

Para medir la efectividad del sistema fue necesario realizar una serie de pruebas con el fin de obtener la información requerida respecto al número de aciertos y desaciertos que el sistema arroja en un entorno real, por ello, se sometieron a inspección 50 piezas, de las cuales, 25 presentaban algún defecto o rasgo característico que el sistema debe ser capaz de identificar, dado que para eso fue entrenado, las otras 25 piezas se encuentran libre de defecto o rasgo identificable.

Los resultados de esta evaluación son representados mediante en una matriz de confusión, la cual tiene como objeto, contrastar los resultados de las predicciones que el sistema realiza en función de los valores reales, con ello, es posible obtener las métricas de Precisión, Sensibilidad (Recall) y el Índice F1 (F1 Score), que nos dan un panorama global del funcionamiento del sistema como se explicó en el capítulo 2.

5.1. Resultados de pruebas realizadas en proceso A

Como se describió en el capítulo 4, el proceso A, corresponde a la empresa Materiales y Molduras de Mexicali, concierne a la manufactura de polín estructural de 3 x 4 pulgadas, fabricado con lámina de acero galvanizado calibre 24, utilizado generalmente en la elaboración de estructuras para distintas aplicaciones, en el que se conformó un set de datos de 180 imágenes para entrenar al modelo de detección, obteniendo los siguientes resultados de la fase de evaluación para el modelo TFLite.

{'AP': 0.25358814,
'AP50': 0.6508635,
'AP75': 0.1303322,
'AP_/bent': 0.18200459,
'AP_/creased': 0.3251717,
'APl': 0.31807923,
'APm': 0.24526039,

'APs': -1.0,
'ARl': 0.44722223,
'ARm': 0.41,
'ARmax1': 0.20484848,
'ARmax10': 0.4078788,
'ARmax100': 0.42454547,
'ARs': -1.0}

Este modelo fue utilizado con el fin de detectar las categorías *"bent"* y *"creased"* en las imágenes captadas desde la cámara integrada del dispositivo móvil, una vez puesta en marcha la aplicación, se dispuso a montar el dispositivo para llevar a cabo la inspección del material, entre las muestras bajo inspección se encontraban piezas libre de defecto, así como piezas con algún defecto visible, esto con el fin de comprobar que la aplicación era capaz de detectar los defectos en las imágenes en tiempo real, a través del flujo de video en el visor de la aplicación.

Figura 5. 1. Maquinaria para la fabricación del material (izquierda). Propuesta de montaje del dispositivo móvil para inspección del material (derecha).

El propósito fue realizar el montaje del dispositivo móvil al final del proceso, esto con el fin de inspeccionar en todo momento, durante la operación de la maquinaria, el

material que está siendo manufacturado, y de esta forma, ser capaz de detectar cualquier defecto que se presente en el proceso de manera oportuna y de ser necesario, detener el proceso para tomar las medidas correctivas necesarias.

Figura 5. 2. Inspección en tiempo real de una muestra de material que presenta defectos visibles causados durante el proceso de manufactura.

Se puede observar en la Figura 5.2 que la aplicación es capaz de detectar los defectos en el material en tiempo real, es posible ajustar el umbral de confianza, lo que permite discriminar aquellas detecciones que se encuentren por debajo de ese umbral, en este caso, se ajustó a *0.70*, lo que significa que solamente se mostraran los recuadros delimitadores de los defectos detectados que se encuentren por encima del 70% de certeza para la detección. La velocidad con la que realiza las detecciones puede variar en función de los hilos de ejecución asignados al procesamiento, en este caso, se configuraron 4 hilos de ejecución, lo que redujo el tiempo de inferencia en el orden de los 200 milisegundos, lo que permite

realizar entre 4 o 5 detecciones por segundo de manera satisfactoria. A continuación, se muestra la matriz de confusión y las métricas obtenidas con los resultados de la prueba realizada en este proceso.

Matriz de confusión

		Valores de Predicción		
		Pieza con defecto	Pieza sin defecto	
Valores Reales	Pieza con defecto	40 (**VP**)	10 (**FN**)	**50**
	Pieza sin defecto	13 (**FP**)	37 (**VN**)	**50**
		53	**47**	**100**

Métricas obtenidas

Precisión	**75.47%**
Sensibilidad (Recall)	**80.00%**
Puntaje F1	**77.66%**

Figura 5. 3. Evaluación del desempeño del sistema en el proceso A.

5.2. Resultados de pruebas realizadas en proceso B

El proceso B, de la empresa Precisión Sheet Metal de México, corresponde a la fabricación de gabinetes elaborados con lamina de acero inoxidable, utilizados para el montaje de quipo electrónico en la industria, principalmente en el sector alimentario, industria del transporte y telecomunicaciones, en el que se conformó un set de datos de 544 imágenes para entrenar al modelo de detección, obteniendo los siguientes resultados de la fase de evaluación para el modelo TFLite.

{'AP': 0.13523853,
'AP50': 0.2501455,
'AP75': 0.10496636,
'APs': -1.0,
'APm': 0.054744177,
'APl': 0.46441644,
'ARmax1': 0.22532609,
'ARmax10': 0.3043478,
'ARmax100': 0.32336956,
'ARs': -1.0,
'ARm': 0.13333334,
'ARl': 0.56875,
'AP_/48117_a': 0.23768976,
'AP_/48117_b': 0.12917817,
'AP_/48117_c': 0.13712871,
'AP_/48118_a': 0.25984574,
'AP_/48118_b': 0.15748569,
'AP_/48118_c': 0.17124758,
'AP_/surface_defect': 0.15321782}

El modelo entrenado fue utilizado en la aplicación con el fin de detectar entre 2 modelos de gabinete (48117 y 48118) de los 8 modelos que la empresa fabrica, de cada modelo, se identificaron los rasgos más significativos, clasificados como *48117_a*, que identifica el tipo de soporte central en el gabinete, *48117_b* para identificar la posición de los remaches a un costado del soporte central y *48117_c* que identifica la posición de los orificios en la parte superior del gabinete. Estos tres rasgos son fundamentales para diferencial entre ambos modelos, por tal motivo, también fueron identificados en el gabinete 48118, con sus respectivos identificadores *48118_a*, *48118_b* y *48118_c*. Una vez puesta en marcha la aplicación se dispuso a montar el dispositivo para llevar a cabo la identificación de los gabinetes, entre las muestras inspeccionadas se encontraban tanto gabinetes tipo 48117 como 48118 para su identificación, así como piezas con acabado superficial que no cumplen con los estándares de calidad, esto con el fin de comprobar que la aplicación es capaz de identificar el modelo al que pertenece cada gabinete, así como detectar defectos en el acabado superficial en las imágenes captadas en tiempo real, a través del flujo de video en el visor de la aplicación.

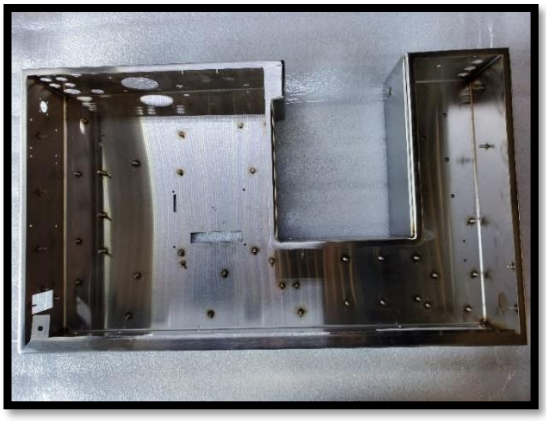

Figura 5. 4. Propuesta de montaje del dispositivo móvil sobre mesa de inspección del material.

La intención primordial fue montar el dispositivo móvil en una posición sobre la mesa de inspección donde fuera posible visualizar en su totalidad cada gabinete, esto con el fin de que la imagen capturada pueda ser procesada por la aplicación para la detección de los rasgos que diferencian entre un modelo de gabinete y otro, y que, a su vez, la aplicación de manera inmediata, mostrara los resultados en pantalla, ayudando en la identificación de cada gabinete con su respectivo modelo.

Figura 5. 5. Monitoreo de manera remota por el operador para la identificación del modelo.

Como se puede observar en la Figura 5.6, el dispositivo fue montado de manera provisional sobre la mesa de inspección, lo que permite al inspector visualizar de manera inmediata en pantalla, el resultado de la detección, y a su vez identificar el modelo de gabinete que se trata, esto sin la necesidad de colocar sobre el gabinete el set de plantillas que ayudan al operador para esta labor de identificación.

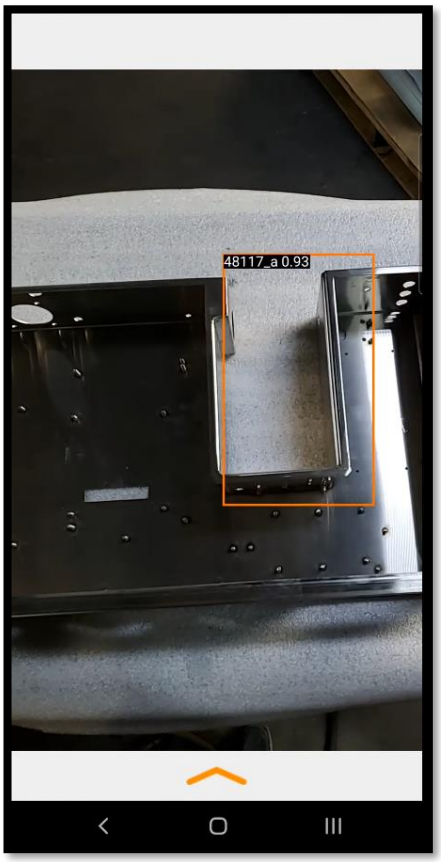

Figura 5. 6. Inspección en tiempo real de gabinete para detección de rasgos característicos.

La aplicación es capaz de detectar los rasgos característicos en las piezas en tiempo real, es posible ajustar el umbral de confianza, lo que permite discriminar aquellas detecciones que se encuentren por debajo de ese umbral, por lo que se ajustó a *0.70*, con el fin de mostrar solamente los recuadros delimitadores de los defectos detectados que se encuentren por encima del umbral de 70% de certeza en la detección. Se ajustaron 4 hilos de ejecución para el procesamiento de las imágenes alcanzando un tiempo de inferencia cercano

a los 200 milisegundos, con detecciones que alcanzaron un umbral del 90% de certeza en algunos casos. A continuación, se muestra la matriz de confusión y las métricas obtenidas con los resultados de la prueba realizada en este proceso.

Matriz de confusión

		Valores de Predicción		
		Pieza con defecto	Pieza sin defecto	
Valores Reales	Pieza con defecto	34 (**VP**)	16 (**FN**)	**50**
	Pieza sin defecto	3 (**FP**)	47 (**VN**)	**50**
		37	**63**	**100**

Métricas obtenidas

Precisión	**91.89%**
Sensibilidad (Recall)	**68.00%**
Puntaje F1	**73.72%**

Figura 5. 7. Evaluación del desempeño del sistema en el proceso B.

5.3. Resultados de pruebas realizadas en proceso C

El proceso C, corresponde al de la empresa Esliter de México, el cual consiste en la fabricación de tubo redondo industrial de acero de 1-5/8 pulgadas de diámetro, elaborado con lamina calibre 18, utilizado en la elaboración de estructuras. En este proceso, se identificaron tres defectos comunes que se hacen ver durante la manufactura del material, por lo que se clasificaron como: *"weld_defect", "weld_joint"* y *"hole",* estos identificadores describen los defectos más comunes en el proceso de fabricación, básicamente, *"weld_defect"* obedece a discontinuidades en la aplicación de la soldadura al unir el material y formal el tubo redondo, *"weld_joint",* se refiere a la presencia de uniones realizadas por

soldadura entre dos segmentos de material y *"hole"* que identifica defectos causados principalmente por una mala configuración del equipo, al desgaste de los rodillos de la maquinaria o al mal funcionamiento del electrodo al momento de realizar la soldadura. Se conformó un set de datos de 430 imágenes para entrenar al modelo de detección, obteniendo los siguientes resultados de la fase de evaluación para el modelo TFLite.

{'AP': 0.29525825,
'AP50': 0.91693985,
'AP75': 0.05570503,
'AP_/hole': 0.32681283,
'AP_/weld_defect': 0.25896195,
'AP_/weld_joint': 0.3,
'APl': 0.30871883,
'APm': 0.24098223,
'APs': -1.0,
'ARl': 0.35833332,
'ARm': 0.3918919,
'ARmax1': 0.2667683,
'ARmax10': 0.38292682,
'ARmax100': 0.40447155,
'ARs': -1.0}

Este modelo se utilizó con el objetivo de detectar las categorías *"weld_defect"*, *"weld_joint"* y *"hole"* en las imágenes captadas desde la cámara integrada del dispositivo móvil, una vez puesta en marcha la aplicación se dispuso a montar el dispositivo para llevar a cabo la inspección del material, entre las muestras bajo inspección se encontraban piezas libre de defecto, así como piezas con algún defecto visible, para que de esta manera fuera posible comprobar que la aplicación era capaz de detectar los defectos en las imágenes en tiempo real, a través del flujo de video en el visor de la aplicación.

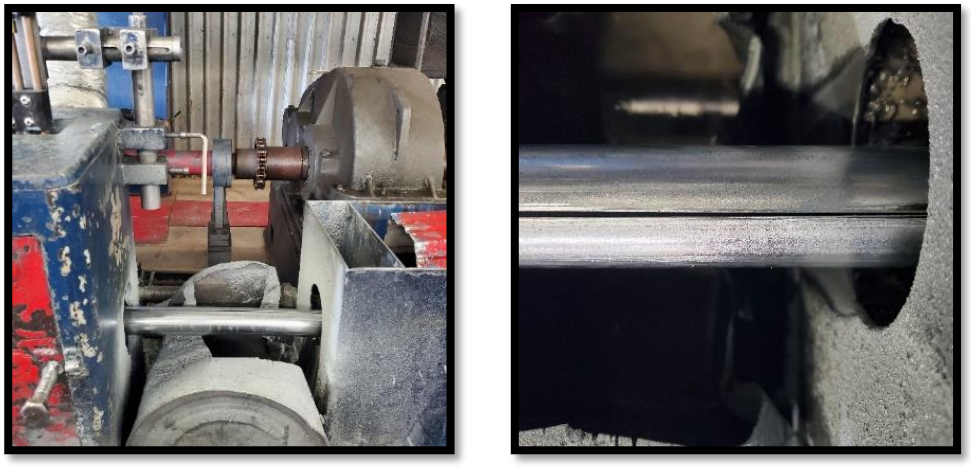

Figura 5. 8. Etapa final del proceso de fabricación del material (izquierda). Propuesta de montaje del dispositivo móvil para inspección del material (derecha).

La idea de montar el dispositivo móvil al final del proceso, es con el propósito de inspeccionar la totalidad del material mientras la maquinaria se encuentra en operación, para de esta forma, ser capaz de detectar cualquier defecto que se presente de manera oportuna, y de ser necesario, detener el proceso para tomar las medidas correctivas necesarias y evitar la producción de piezas con el defecto identificado.

La detección oportuna es crítica en este proceso, ya que no se cuenta con un mecanismo para identificar si el material que se está manufacturando se encuentra libre de defectos. En esta última etapa del proceso, el tubo es cortado generalmente en tramos de 20 pies de longitud o alguna otra medida indicada por el fabricante, posteriormente, una vez cortado, es transportado a una sección donde se empaca, para finalmente ser despachado.

Figura 5. 9. Corte del material al finalizar el proceso (izquierda). Banda transportadora del material hacia el área de empacado (derecha).

Dado que esta última etapa del proceso se realiza de manera automatizada, no es hasta el momento en que el material se despacha, cuando el operador tiene contacto directo con este, y es en ese instante cuando se notifica que el material presenta algún defecto y es momento de detener el proceso y realizar las medidas correctivas necesarias en la maquinaria para identificar el origen de la falla que ocasiona este desperfecto.

Figura 5. 10. Inspección en tiempo real de una sección de tubo que presenta defecto de fabricación.

La aplicación puesta en marcha, fue capaz de detectar los defectos que se presentaron en el material de forma satisfactoria. Con el fin de mejorar los resultados, es posible ajustar el umbral de confianza, lo que permite discriminar aquellas detecciones que se encuentren por debajo de ese umbral, en este caso se ajustó a *0.70*, lo que permite mostrar únicamente los recuadros delimitadores de los defectos detectados que se encuentren por encima del umbral de 70% de certeza en la detección. Se ajustaron 4 hilos de ejecución para el procesamiento de las imágenes alcanzando un tiempo de inferencia cercano a los 100 milisegundos, lo que permite realizar entre 8 y 10 detecciones por segundo. El porcentaje de certeza de las detecciones fue de alrededor del 80%, en algunos casos por encima de este valor. A continuación, se muestra la matriz de confusión y las métricas obtenidas con los resultados de la prueba realizada en este proceso.

Matriz de confusión

		Valores de Predicción		
		Pieza con defecto	Pieza sin defecto	
Valores Reales	Pieza con defecto	42 (**VP**)	8 (**FN**)	**50**
	Pieza sin defecto	9 (**FP**)	41 (**VN**)	**50**
		51	**49**	**100**

Métricas obtenidas

Precisión	**82.35%**
Sensibilidad (Recall)	**84.00%**
Puntaje F1	**83.16%**

Figura 5. 11. Evaluación del desempeño del sistema en el proceso C.

CAPÍTULO 6
DISCUSIONES Y CONCLUSIONES

Pruebas en proceso A

Para el estudio de este proceso, el set de datos se conformó solamente de 180 imágenes, ya que el número de muestras de material era limitado al momento del estudio, sin embargo, de las imágenes obtenidas, algunas presentaban defectos fácilmente identificables, y en otras, los defectos eran ligeramente observables, lo que originó que al momento del procesamiento de las mismas, algunos rasgos fueran más difícil de identificar. Aun así, se lograron resultados aceptables, ya que, de acuerdo a las métricas obtenidas en la experimentación, el sistema fue capaz de detectar los defectos con una precisión del 75% y una sensibilidad del 80%, estos datos nos indican que el sistema es efectivo en identificar los rasgos 8 de cada 10 veces y cuando lo realiza, lo hace con una precisión del 75%, lo que es aceptable considerando que el umbral mínimo aceptado fue ajustado al 70%. Sin embargo, estos resultados pueden ser mejorados en una siguiente etapa de la investigación, añadiendo más muestras al set de datos, tomando muestras en diversas perspectivas, así como apoyándose de técnicas para el aumento de datos, con el fin de crear un set de datos robusto que mejore de manera significativa las capacidades de detección de modelo.

Pruebas en proceso B

Al identificar las necesidades de este proceso, se encontró con dos objetivos primordiales, por una parte, se buscó que la aplicación fuera capaz de clasificar cada uno de los distintos modelos de gabinete, que como se mencionó en los apartados anteriores, la empresa fabrica 8 modelos distintos, cada uno con rasgos muy particulares que lo hacen diferenciarse entre los otros modelos. En general, la apariencia física de estos es muy semejante y difícilmente identificable para un operador poco experimentado, por lo que la aplicación para este propósito cumplió el objetivo propuesto.

El segundo objetivo fue el de identificar marcas ocasionadas por la maquina lijadora encargada de dar el acabado final, fue en este aspecto donde se presentaron algunas dificultades. Por una parte, no se contaba con muestras suficientes para evidenciar el mal acabado en los gabinetes, del set de datos conformado de 544 muestras, solamente el 15 %

pertenecían a la clase *"surface_defect"*, lo que ocasionó que el entrenamiento del modelo para este propósito no fuera lo suficientemente robusto, dificultando la labor de detección para este tipo de defecto. Sumando a ellos que, al tratarse de una superficie lustrosa al ser fabricados de acero inoxidable, las condiciones de luz y el reflejo en la superficie juegan un papel adverso al momento de intentar realizar la detección en la superficie. Los resultados obtenidos de la experimentación mostraron que el sistema tiene una precisión de 91%, lo que es aceptable, sin embargo, su sensibilidad (recall) fue de tan solo el 68%, valor por debajo del umbral definido como aceptable, estas métricas nos indican que el sistema es muy poco sensible, ya que tiene dificultades para identificar los rasgos, pero cuando lo logra, lo hace con una muy buena precisión por encima del 90%

Como se mencionó anteriormente, para detectar de manera más efectiva dichos defectos, se pueden usar técnicas de aumento de datos o intensificadores de datos basados en incrementar el número de elementos existentes para aumentar el número de imágenes disponibles. El objetivo de esta práctica es entrenar al sistema con posibles variaciones que pueda encontrar el detector de imágenes que atenten contra la estabilidad del mismo. Adicionalmente, el mantener las condiciones de iluminación lo más similares posible, tanto al momento de realizar la captura de las imágenes de muestra para entrenamiento, así como mantener las mismas condiciones de iluminación en el ambiente donde se llevarán a cabo las detecciones en campo mejorarán significativamente los resultados.

Pruebas en proceso C

En el estudio de este proceso, se identificaron necesidades importantes en cuanto a la capacidad de detección de defectos que llegan a presentarse en el material durante su fabricación, esto debido a que no se cuenta con un sistema de control de calidad que monitorea constantemente el producto que sale de la línea de producción, lo que ocasiona que los defectos sean detectados en la etapa del embalaje, momento en el cual la maquinaría continua en operación, por lo que sigue manufacturando piezas que presentan el mismo defecto.

Para la creación del set de datos, se contaba con suficientes muestras del material para generar una colección de 430 imágenes, lo suficientemente robusto para que las detecciones se efectúen satisfactoriamente. Los resultados obtenidos de la experimentación mostraron que el sistema tiene una precisión del 82% y una sensibilidad del 84%, este equilibrio entre sus métricas le permite detectar los defectos con una precisión por encima del 80% en la mayoría de las ocasiones, sin embargo, el set de imágenes solo consideró material tubular de acero de 1-5/8 pulgadas de diámetro y la maquinaria puede ser configurada para trabajar con diferentes dimensiones, motivo por el cual, sería necesario añadir a la colección de datos, un considerable número de muestras de material con todas las dimensiones disponibles, esto con el fin de que el modelo sea capaz de identificar defectos en todas las variantes del producto.

6.1. Conclusión general

De forma general, se puede concluir que el sistema de detección propuesto en esta investigación fue capaz de cumplir con su propósito primordial, que es la detección oportuna en tiempo real de rasgos en imágenes captadas por un dispositivo móvil, detecciones que son de interés para un propósito en particular.

De acuerdo a las necesidades de los procesos estudiados, el objetivo fundamental fue el detectar defectos que se presentan durante las diversas etapas del proceso de fabricación y que pueden ser identificables a simple vista. A continuación, se muestra un concentrado de las métricas obtenidas conforme a la experimentación en cada proceso.

Tabla 6- 1. Concentrado de las métricas obtenidas para los procesos evaluados.

	Precisión	**Sensibilidad (Recall)**	**Índice F1 (F1 Score)**
Proceso A	75.47	80	**77.66**
Proceso B	91.89	68	**73.72**
Proceso C	82.35	84	**83.16**

Como se puede apreciar en la Tabla 6-1, en el proceso A, el sistema presentó una precisión del 75%, lo que es baja en comparación con los otros procesos, sin embargo, su sensibilidad fue del 80%, lo que le permite realizar detecciones de manera moderada a una precisión baja, pero aun aceptable, considerando el umbral mínimo del 70% propuesto al inicio de la investigación.

En el proceso B, el sistema logró detecciones con una precisión superior al 90%, un porcentaje aceptable, pero en contraste su sensibilidad fue del 68%, lo que es significativamente bajo. Estas métricas nos indican que el modelo tiene dificultades para detectar los defectos o rasgos de interés para el que fue entrenado, pero cuando logra la detección, lo hace con una muy buena precisión, sobrepasando el 90% de certeza en su detección.

Por último, el desempeño del sistema en el proceso C, presentó un equilibrio en sus métricas, ya que su precisión promedio fue del 82% con una sensibilidad del 84%, lo que le permite detectar los rasgos a un nivel de confianza aceptable en la mayoría de las ocasiones, de acuerdo a los criterios establecidos en esta investigación.

Finalmente, los tres procesos lograron obtener un Índice F1 por encima del 70%, propuesto como umbral mínimo para propósitos de la aplicación desarrollada, este índice es una métrica ampliamente utilizada para la evaluación de modelos de aprendizaje ya que resume la precisión y sensibilidad (recall) en un solo valor numérico.

Un punto importante a resaltar, es que el modelo puede ser entrenado para atender otras necesidades, y en consecuencia, es posible ampliar su uso en otras aplicaciones de la industria, que van desde la detección de rasgos en diversos tipos de materiales, como polímeros, cerámicos, materiales compuestos, componentes electrónicos, y no solo en superficies metálicas como fue el objeto de estudio en esta investigación. Además de brindar apoyo en las labores de inspección visual en otro tipo de procesos, donde se requiere identificar aspectos visibles a la vista del inspector.

Como todo desarrollo, la posibilidad de mejora siempre queda latente, la capacidad de los modelos de detección disponibles, las herramientas para gestionar los datos, la

innovación de nuevas herramientas, así como aquellos recursos de cómputo que hacen posible la creación de aplicaciones enfocadas a este propósito, evolucionan constantemente, lo que da pie a la mejora continua y al desarrollo de nuevas tecnologías que son de gran utilidad para satisfacer necesidades tanto en el sector industrial, como en nuestra vida daría.

Bibliografía

[1] J. Jarvinen and J. Rauhamaa, “Real time surface inspection of steel strips,” *Machine vision news*, vol. 7, Jan. 2004.

[2] B. G. and D. W. B. Batchelor, *Commercial vision systems, in Computer Vision: theory and industrial Applications*. New York: Springer-Verlag, 1992.

[3] Isra Vision, “Surface Vision. Optical in-line inspection at highest speeds.,” 2021. http://www.isravision.com (accessed Sep. 29, 2021).

[4] K. Saitwal, A. A. Maciejewski, and R. G. Roberts, “A Comparison of EigenDecomposition for Sets of Correlated Images at Different Resolutions,” in *Conference on Intelligent Robots and Systems*, 2003, pp. 1011–10.

[5] M. Grijalvo, “La calidad en el sector aeroespacial: Normativa y esquema de certificación,” 2005. [Online]. Available: https://www.researchgate.net/publication/45161241

[6] D. Gustavo and E. Espinosa, “Diseño y construcción de un equipo para la extrusión directa, inversa y mixta, en frío de perfiles metálicos no ferrosos para el laboratorio de procesos de manufactura del DECEM,” Escuela Politécnica Del Ejército, 2012.

[7] C. Alejandro, P. Andrade, D. Marcelo, and T. Vilaña, “Diseño y construcción de laminadora de cuatro rodillos para laminar tiras de aluminio,” Escuela Politécnica Nacional, 2014.

[8] Isra Vision, “Parsytec suface inspection system,” 2021. http://www.simply-inspection.com/index.php?id=48 (accessed Sep. 29, 2021).

[9] C. Roncancio Valencia, F. Gayubo Rojo, J. Gómez García Bermejo, and E. Zalama Casanova, “Detección e identificación de defectos superficiales en diversas clases de chapa laminada mediante visión por computador y redes neuronales,” 2018.

[10] J. Bunge, “Ensayos no destructivos,” Jun. 2011.

[11] F. J. Carrión, V. María, G. Lomelí González, J. Antonio, Q. Rodríguez, and M. M. Madrid, *La evaluación no destructiva de materiales estructurales y puentes*. 2003.

[12] R. R. David, “Desarrollo y elaboración del manual de procedimientos de inspección para talleres aeronáuticos de reparación de ensayos no destructivos,” Institución Universitaria Los Libertadores, Bogotá, 2016.

[13] D. Balageas, “Taking into account heat losses in front-face pulse stimulated thermography experiments View project Application on THZ imaging View

project," 2005. [Online]. Available: https://www.researchgate.net/publication/234038055

[14] D. Gauna *et al.*, "Ensayos no destructivos," 2012.

[15] C. Fernando and B. Reinoso, "Estudio técnico e implementación del laboratorio de ensayos no destructivos (END) para el área de ciencia y tecnologías de la Universidad Politécnica Salesiana Sede Cuenca," Universidad Politécnica Salesiana Sede Cuenca, 2014.

[16] J. G. Guerrero, "Inspección en la aeronáutica," 2017.

[17] Naina Khedekarmay, "KPCB estimates based on publicly disclosed company data," *KPCB*, May 29, 2014. https://www.firstpost.com/tech/news-analysis/now-upload-share-1-8-billionphotos-everyday-meeker-report-3652169.html (accessed Sep. 29, 2021).

[18] Jae Duk Seo, "Understanding and Visualizing Convolutional Neural Networks," *Medium*, May 28, 2019. https://medium.com/@SeoJaeDuk/archived-post-understanding-and-visualizingconvolutional-neural-networks-d6da3e2851cf (accessed Sep. 29, 2021).

[19] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew., "Deep learning for visual understanding: A review," 2016.

[20] K. Chen, T. Mikolov, and I. Sutskever, "Distributed representations of words and phrases and their compositionality," 2013.

[21] G. E. Hinton, A. Krizhevsky, and I. Sutskever., "Imagenet classification with deep convolutional neural networks," 2012.

[22] J. Schmidhuber, D. Ciresan, and U. Meier, "Multi-column deep neural networks for image classification," 2012.

[23] J. Schmidhuber, D. C. Ciresan, and U. Meier, "Transfer learning for latin and chinese characters with deep neural networks," 2012.

[24] L. Xu and J. S. J. Ren, "On vectorization of deep convolutional neural networks for vision tasks," 2015.

[25] Bhiksha Raj and Haohan Wang, " On the origin of deep learning," 2017.

[26] S. Osindero, G. E. Hinton, and Y. W. Teh, "A fast learning algorithm for deep belief nets, neural computation," 2006.

[27] JS Denker, D Henderson, RE Howard, W Hubbard, Y LeCun, and B Boser, "Backpropagation applied to handwritten zip code recognition," 1989.

[28] Sumit Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,” 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neuralnetworks-the-eli5-way-3bd2b1164a53 (accessed Sep. 29, 2021).

[29] Grace W. Lindsay, “Convolutional neural networks as a model of the visual system: Past, present, and future,” 2020.

[30] Shou-tao Xu, Xindong Wu, Zhong-Qiu Zhao, and Peng Zheng, “Object detection with deep learning: A review,” 2019.

[31] Course Website, “Convolutional Neural Networks (CNNs / ConvNets),” 2018. https://cs231n.github.io/convolutional-networks/ (accessed Sep. 29, 2021).

[32] J. Denker, Y. LeCun, and B. Boser., “Handwritten digit recognition with a backpropagation network, advances in neural information processing systems,” 1990.

[33] Yangqing Jia Pierre *et al.*, “Going deeper with convolutions,” 2015.

[34] Shaoqing Ren, Kaiming He, Xiangyu Zhang, and Jian Sun, “Deep residual learning for image recognition,” 2015.

[35] Q. Yang and S. J. Pan, “A survey on transfer learning, ieee transactions on knowledge and data engineering,” 2010.

[36] Shaoqing Ren, Kaiming He, Xiangyu Zhang, and Jian Sun, “Deep residual learning for image recognition,” 2015.

[37] Dipanjan (DJ) Sarkar, “A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning,” 2018. https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transferlearning-with-real-world-applications-in-deep-learning-212bf3b2f27a (accessed Sep. 29, 2021).

[38] Jonathan Hui, “What do we learn from region based object detectors (Faster R-CNN, R-FCN, FPN)?,” 2018. https://jonathan-hui.medium.com/what-do-we-learn-from-region-based-object-detectors-faster-r-cnn-r-fcn-fpn-7e354377a7c9 (accessed Nov. 04, 2021).

[39] Theo Gevers Arnold W. M., Smeulders Koen E. A., van de Sande, and Jasper R. R. Uijlings, “Segmentation as selective search for object recognition,” 2011.

[40] Trevor Darrell, Ross B. Girshick, Jeff Donahue, and Jitendra Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation.,” 2013.

[41] Ross B. Girshick, “Fast r-cnn,” 2015.

[42] Ross B. Girshick, Shaoqing Ren, Kaiming He, and Jian Sun, “Faster r-cnn: towards real-time object detection with region proposal networks,” 2015.

[43] Kaiming He, Jifeng Dai, Yi Li, and Jian Sun, “R-fcn: object detection via region-based fully convolutional networks,” 2016.

[44] W. Liu *et al.*, “Single shot mutlibox detector,” 2015.

[45] Jonathan Hui, “SSD object detection: Single Shot MultiBox Detector for real-time processing,” 2018, Accessed: Nov. 04, 2021. [Online]. Available: https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06

[46] Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, Mark Sandler, and Andrew Howard, “Mobilenetv2: Inverted residuals and linear bottlenecks,” 2019.

[47] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, “You only look once: Unified, real-time object detection,” 2015.

[48] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, “Yolov4: Optimal speed and accuracy of object detection,” 2020.

[49] Joseph Redmon and Ali Farhadi, “Yolov3: An incremental improvement,” 2018.

[50] He Chang-Wei Huang, Liqing Wei, Lingling Li, and Guo Anfu, “Tf-yolo: An improved incremental network for real-time object detection,” *Applied Sciences*, 2019.

[51] Jonathan Hui, “mAP (mean Average Precision) for Object Detection,” 2018, Accessed: Nov. 04, 2021. [Online]. Available: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

[52] Profesor Data, “La Matriz de Confusión: ¡claramente explicada!,” 2022. https://profesordata.com/2020/08/07/evaluando-los-modelos-de-clasificacion-en-aprendizaje-automatico-la-matriz-de-confusion-claramente-explicada/ (accessed Oct. 30, 2022).

[53] Gartner Group, “Gartner Group,” 2018. https://www.gartner.com/en (accessed Sep. 22, 2022).

[54] Roberto Adeva, “Qué es Android: todo sobre el sistema operativo de Google,” 2022. https://www.adslzone.net/reportajes/software/que-es-android/ (accessed Oct. 30, 2022).

[55] Android Developers, “Arquitectura de la plataforma,” 2017. https://developer.android.com/guide/platform/index.html?hl=es (accessed Sep. 22, 2022).

[56] Rocío GR, "¿Qué es iOS? Todo sobre el sistema operativo de Apple," 2022. https://www.adslzone.net/reportajes/software/que-es-ios/ (accessed Oct. 30, 2022).

[57] Javier Cala Uribe, "Guía iOS: Desarrollando aplicaciones para dispositivos móviles," 2010. http://www.maestrosdelweb.com/guia-desarrollo-iphone-ipad/ (accessed Oct. 30, 2022).

[58] Blogger, "Sistema Operativo iOS," 2017. http://ios-sistema.blogspot.com/p/arquitectura-ios.html (accessed Oct. 30, 2022).

[59] E. Arcos, "El silencioso final de Windows 10 Mobile," 2019. https://hipertextual.com/2019/01/silencioso-final-windows-10-mobile (accessed Oct. 30, 2022).

[60] Microsoft, "Fin del soporte para Windows 10 Mobile en 2019," 2022. https://learn.microsoft.com/es-es/lifecycle/announcements/windows-10-mobile-end-of-support (accessed Sep. 22, 2022).

[61] Carlos Luis Murillo, "Windows Phone 8," 2015.

[62] Google, "How to use Play Console," 2022. https://support.google.com/googleplay/android-developer/answer/6112435?hl=en#zippy= (accessed Sep. 22, 2022).

[63] Tensorflow, "Tensorflow plataforma para el aprendizaje automático," 2022, Accessed: Sep. 22, 2022. [Online]. Available: https://www.tensorflow.org/overview

[64] Google, "Colaboratory - Conceptos básicos," 2022. https://research.google.com/colaboratory/intl/es/faq.html#:~:text=Colaboratory%2C%20o%20%22Colab%22%20para,an%C3%A1lisis%20de%20datos%20y%20educaci%C3%B3n. (accessed Sep. 22, 2022).

[65] Izary Rondón, "¿Qué es Anaconda?," 2022. https://eiposgrados.com/blog-python/que-es-anaconda/ (accessed Sep. 22, 2022).

[66] Aitor Alcázar Fernández, "Desarrollo de un sistema automático de reconocimiento de lenguaje de signos," 2019.

[67] Tensorflow, "Creador de modelos TensorFlow Lite," 2022. https://www.tensorflow.org/lite/models/modify/model_maker (accessed Sep. 22, 2022).

[68] R. P. Q. V. le Mingxing Tan, "EfficientDet: Scalable and Efficient Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[69] Tensorflow, “Model Maker - create,” 2022. https://www.tensorflow.org/lite/api_docs/python/tflite_model_maker/object_detector/create (accessed Sep. 22, 2022).

[70] Rajdeep Dua, *Mastering TensorFlow 2.x: Implement Powerful Neural Nets across Structured, Unstructured datasets and Time Series Data*. 2022.

[71] Casa Robles Paulo Cesar, “Desarrollo de aplicaciones móviles de clasificación y detección de objetos a partir de redes convolucionales ligeras,” 2020.